

**Oracle[®] Communications Messaging Server
Reference**

Release 8.0.1

E64178-01

September 2015

Oracle[®] Communications Messaging Server Reference

Release 8.0.1

E64178-01

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Preface	v
I Configuration syntax	1
1 Option value syntax	1-1
2 Options for migrating to Unified Configuration	2-1
3 Special symbolic names	3-1
4 Recipe language	4-1
5 Sieve filters	5-1
6 TCP wrappers	6-1
II Infrastructure	1
7 Base options	7-1
8 Scheduler options	8-1
9 Watcher options	9-1
10 msprobe options	10-1
11 Alarm options	11-1
12 Auth options	12-1
13 sectoken options	13-1
14 Deployment Map options	14-1
15 rollovermanager options	15-1
III The Message Store	1
16 Message Store options	16-1
17 message_language options	17-1
18 Partition options	18-1
19 backup_group options	19-1
20 Client access to Message Store servers	20-1
21 IMAP options	21-1
22 POP options	22-1
23 Message Trace options	23-1
24 notifytarget options	24-1
25 Indexer options	25-1
IV Proxies and the MMP	1
26 Proxy options	26-1
27 MMP and IMAP Proxy and POP Proxy and SUBMIT Proxy and vdomain options	27-1
28 tcp_listen options	28-1
V Convergence webmail	1
29 MSHTTP options	29-1
30 SMIME options	30-1
31 SSO options	31-1
32 icapservice options	32-1
VI The MTA	1
33 Channels	33-1
34 Rewrite rules	34-1
35 Aliases	35-1
36 Mailing lists	36-1
37 Mapping tables	37-1
38 Message conversions	38-1
39 MTA options	39-1
40 MTA Tailor options	40-1
41 Dispatcher	41-1
42 Job Controller	42-1

43 Compiling the MTA configuration	43-1
44 Mail filtering and access control	44-1
45 Spam and virus filtering	45-1
46 MeterMaid	46-1
47 Notification messages	47-1
48 Message tracking and recall	48-1
49 TCP/IP channels	49-1
50 BSMTP channels	50-1
51 ims-ms channels	51-1
52 Other channels	52-1
53 SMS options	53-1
54 Message capture	54-1
55 Monitoring the MTA	55-1
56 MTA performance tuning	56-1
57 Restricting information emitted	57-1
58 MTA command line utilities	58-1
VII Additional components	1
59 PAB options	59-1
60 SNMP options	60-1
61 ENS options	61-1
62 eval_ldapd options	62-1
A Supported Standards	A-1
Glossary	G-1
Index	Index-1

Preface

This technical reference manual documents the various options and facilities provided by Oracle Communications Messaging Server.

The preface covers the following:

- Audience
- Documentation Accessibility
- Related Documents

1.1 Audience

This document is intended for Messaging Server administrators and developers who want to configure and manage their Messaging Server infrastructure.

1.2 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>

1.3 Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

1.4 Related Documents

For more information, see the following documents in the Messaging Server documentation set:

- *Messaging Server Installation and Configuration Guide*
- *Messaging Server Release Notes*
- *Messaging Server Security Guide*
- *Messaging Server System Administrator's Guide*

Part I Configuration syntax

In Unified Configuration, nearly all configuration options are stored in the `config.xml` Message Server unified configuration file, as XML elements; a few, security-related, options (*e.g.*, passwords) are stored in `restricted.cnf`. However, under normal circumstances, the Messaging Server unified configuration file `config.xml` is not---indeed should not be--- inspected or edited manually by the Messaging Server administrator. Instead, normally Messaging Server's `msconfig` utility is used to examine the configuration and make configuration changes.

Options in the unified configuration file `config.xml` generally are typed XML elements. The `msconfig` utility performs type checking on configuration settings it makes. See [Option value syntax](#) for further details.

The `instancename` and `rolename` options discussed in [Options for migrating to Unified Configuration](#) set the context for option values.

Sets of commands for `msconfig` may be scripted using the [Recipe language](#).

A few [special symbolic names](#) may be used in option values, or in the [Recipe language](#).

Another language used by multiple components of Messaging Server, including the [MTA](#) and the [Message Store \(for purge operations\)](#), as well as by external components such as some email user agents, is the [Sieve language](#).

Several components of Messaging Server make use of the [TCP wrapper](#) concept, for access controls.



Chapter 1 Option value syntax

1.1 Available Types	1-1
1.2 ISO 8601 P format	1-3
1.3 ISO 8601 format	1-3
1.4 MTA URL types	1-4
1.4.1 LDAP URL substitution sequences	1-5

Option settings in Unified Configuration, that is, values in the file `config.xml`, generally are typed XML elements. The `msconfig` utility performs type checking on the configuration settings it makes, and the `msconfig` utility will issue an error if an attempt is made to set an invalid value. The immediate validation and feedback the `msconfig` utility provides on configuration option settings is one of the advantages available by using the Unified Configuration. In contrast, with a legacy configuration, errors in option setting syntax or option values might not be reported until a process attempted to execute; for instance in the case of the MTA, perhaps not until an `imsimta cnbuild` or similar command were issued.

Note: The errors reported by the `msconfig` utility are typically *much* easier to understand and hence correct than the general XML errors that would be reported by an XML validation of `config.xml`. This is a significant reason why it is important to use `msconfig` to inspect and modify the MTA configuration, rather than attempt to modify `config.xml` directly.

The `msconfig` utility can show the type of any option, and the default value, if any, using the `-type` switch and `-default` switch, respectively; for instance:

```
# msconfig
msconfig> show mta.enable -type
mta.enable>: bool
msconfig> show mta.enable -default
mta.enable: 0
```

The `msconfig` interactive help text for an option may also provide additional guidance on proper values for an option; for instance,

```
msconfig> help option enable
```

The `msconfig` utility knows the type permitted for each option and will issue a (reasonably clear) error if an attempt is made to set an invalid value. For instance:

```
msconfig> set mta.enable "localhost"
Error setting option mta.enable: Option value is not a valid value for the option (-23)
```

1.1 Available Types

For the underlying `config.xml` Unified Configuration, quite a few different XML types are defined and can potentially be declared as valid for various option values, including but not necessarily limited to (as this list can be expected to grow) those shown below. Note that NUL characters are not allowed in string types. Note also that all list types are space separated lists; in particular, CRs and LFs are not allowed, leading and trailing spaces are not allowed, and runs of two or more (unquoted) spaces are not allowed.

- *String and character types*

- UTF8 string
- UTF8 character
- UTF8 text node
- String (UTF8)
- Non-empty string (UTF8)
- ASCII string
- Non-empty ASCII string
- ASCII character
- Printable ASCII string
- Printable ASCII character
- Printable ASCII string list
- Enumerated string case-sensitive
- Enumerated string case-insensitive
- Name
- Name list
- URL

- *Numeric types*
 - 32 bit integer
 - List of 32 bit integers
 - Unsigned 32 bit integer
 - List of 32 bit unsigned integers
 - Unsigned 64 bit integer
 - Unsigned 16 bit integer
 - List of unsigned 16 bit integers
 - Boolean
 - Boolean true-only
 - Floating point
 - Unsigned octal (maximum length 9 octal digits)
 - Enumerated 32 bit integer values

- *Time types*
 - ISO 8601 time
 - List of ISO 8601 times
 - ISO 8601 duration
 - List of ISO 8601 durations
 - ISO 8601 duration OR time

- *Host, domain, and IP types*
 - Domain
 - Domain and port
 - Host
 - Host and port
 - IPv4
 - IPv4 list
 - IPv6
 - IPv6 list
 - IPv4 literal
 - Domain literal list
 - IPv4 and port
 - Host list
 - IPv4 range

- *File and directory types*

- MTA-specific directory path
- MTA-specific file path
- MTA-specific path list
- Directory path
- Absolute directory path
- File path
- Relative path
- Path
- File name
- *Password type*
- *LDAP types*
 - LDAP URL
 - LDAP attribute name
 - LDAP DN
- *Address types*
 - [RFC 822](#) address
- *Bit mask*
- *Various enumerated types*

1.2 ISO 8601 P format

The ISO 8601 P, or ISO 8601 duration, format is:

PyearMonthMweekWdayDThourHminuteMsecondsS

where the values *year*, *month*, *etc.*, are integer values specifying a duration or an offset (delta) from the current time. The initial P is required; other fields may be omitted, though the T is required if any time values are specified.

For example, PT1H means a one hour duration or offset.

Besides the [backoff](#) channel option and a few [alias options](#), also several [MeterMaid options](#) take ISO 8601 P arguments.

1.3 ISO 8601 format

The ISO 8601, or ISO 8601 time, format is either specified in Greenwich Mean time (GMT or Zulu):

yyyy-mm-ddThhmmss.ssZ

or with an optional time zone offset:

yyyy-mm-ddThhmmss.ss+hh:mm

or

yyyy-mm-ddThhmmss.ss-hh:mm

The hyphens in the date portion are optional and may be omitted; though for a negative time zone offset, a hyphen/minus must of course be specified. Spaces are ignored. Year is specified

in four digits, each of month, day, hours, and minutes is specified in exactly two digits (using a leading zero for values less than 10), and seconds is typically specified in exactly two digits (using a leading zero for values less than 10) or optionally can include a decimal point followed two more digits specifying hundredths of a second. Hours are specified on a 24 hour clock. For instance:

```
2013-05-22T12:30:00-08:00
```

Note that prior to Messaging Server 7.0, only a somewhat more restricted format was supported: hyphens were not permitted in the date, nor was a time zone offset permitted (Z was required), nor were fractions of a second (hundredths of a second) permitted.

1.4 MTA URL types

A number of [MTA options](#), [channel options](#), [alias options](#), *etc.*, allow values of various URL types. Such URL types may include:

- `file:` -- used to refer to files stored in the local filesystem
- `ldap:` -- used to refer to data stored in the LDAP directory
- `ldaps:` -- used to refer to data stored in the LDAP directory, accessed via LDAP+SSL
- `pabldap:` -- used to refer to data stored in a Personal Addressbook LDAP directory
- `pabldaps:`
- `extldap:`
- `extldaps:`
- `ssrd:`
- `ssr:` -- access [Sieve filter](#) stored in deprecated [Server Side Rules database](#)
- `mailto:`
- `data:` -- URIs make it possible to specify data directly, in the URI itself
- `http:`
- `imap:` -- access data using IMAP. The credentials specified by the [imap_username](#) and [imap_password](#) MTA options are used to log in to the IMAP server and the URL is resolved with the URLFETCH IMAP command. Note that `imap:` URL resolution is part of the server-side support for the [BURL SMTP extension](#) (used to implement forwarding of messages without having to download them) so any usage of such URLs by the MTA must take into account the fact that there's only one set of such login credentials.
- `imaps:` -- access data using IMAP+SSL.
- `metermaid:`
- `memcache:` -- new in MS 8.0

In addition, some such options may also support a non-URL form argument, assumed to be of an especially "appropriate" type for that option, depending upon the option. For instance, some options assume a file-path argument when no URL prefix is present, while other options might assume an e-mail address argument when no URL prefix is present.

Some [MTA options](#), [channel options](#), or [alias options](#) may allow use of certain substitution sequences in their value settings. The most general list of such substitution sequences may be found in the discussion of [LDAP URL substitution sequences](#). However, some (but only some!) of these substitution sequences are also valid in other types of URL settings for certain MTA option or channel option or alias option settings. See discussion of specific options for details.

Note that in order to use as option values forms of URL that involve querying some external-to-the-MTA component, such as an LDAP URL, or the special "Personal Addressbook" (`paldap:`) form of URL, or an IMAP (BURL) URL, or a MeterMaid URL, or a Memcache URL, typically configuration of just *how* the MTA should connect to that other component is necessary -- that is, configuration of just how to properly interpret such values is necessary. See MTA options such as the [LDAP bind and connect MTA options](#), [LDAP PAB MTA options](#), [BURL MTA options](#), [Memcache MTA options](#), or [MeterMaid MTA options](#) for performing such configuration.

1.4.1 LDAP URL substitution sequences

When specifying LDAP URLs for MTA use, various substitution sequences, as shown in [Table of LDAP URL substitution sequences](#), are generally available.

Table 1.1 LDAP URL substitution sequences

Substitution Sequence	Description
\$\$	Literal \$ character
\$~account	Home directory of user account
\$?string?	(New in 8.0) Apply the Message Store's <code>hashdir</code> algorithm to <code>string</code> to produce a directory path
\$\	Force subsequent material to lower case
\$^	Force subsequent material to upper case
\$_	Leave case as-is for subsequent material
\$ /table-name/argument	Call out to mapping table <code>table-name</code> , probing with <code>argument</code> ; if the mapping table is found an a \$Y, \$y, \$T, or \$t is returned, then use the returned string. Note that the slash shown before and after the table-name can in fact be any character; a character should be used that doesn't conflict with the expected characters in either table-name or argument.
\$A	Address
\$nA	Insert the <i>n</i> th character of the Address
\$B	LDAP user root; <i>i.e.</i> , the value of the ugldapbasedn base option (in legacy configuration, the <code>local.ugldapbasedn</code> configutil parameter), or as overridden by the MTA-specific ldap_user_root MTA option
\$C	LDAP domain root; <i>i.e.</i> , the value of the dcroot base option (in legacy configuration, the <code>service.dcroot</code> configutil parameter), or as overridden by the MTA-specific ldap_domain_root MTA option
\$D	Domain name

\$E	Synonymous with \$1E; that is, substitute in the value of the LDAP attribute named by ldap_spare_1
\$nE	Substitute in the value of the LDAP attribute named by the <code>ldap_spare_n</code> MTA option, where <i>n</i> is in the range 1-18 (any other value is equivalent to 1); note that ldap_spare_6 was added for Messaging Server 7.0-3.01; ldap_spare_7 through ldap_spare_18 were added for Messaging Server 7.2-7.02.
\$F	Delivery filename; some syntax checking is done if the "name" is a file URL (<code>file:</code> syntax)
\$G	Synonymous with \$2G; that is, substitute in the value of the LDAP attribute named by ldap_spare_2
\$nG	Similar to \$nE, but with the default for <i>n</i> being 2 rather than 1
\$H	Host name (first portion of fully qualified domain name)
\$I	Host domain (the canonical domain name for the domain the user is in), unless the host domain is the same as the defaultdomain option (in legacy configuration, the <code>service.defaultdomain</code> configutil parameter), or the MTA-specific override option ldap_default_domain ; when the host domain matches the <code>defaultdomain</code> , then a null string is substituted
\$1I	Host domain (the canonical domain name for the domain the user is in), unless the host domain is the same as the defaultdomain option (in legacy configuration, the <code>service.defaultdomain</code> configutil parameter), or the MTA-specific override option ldap_default_domain ; when the host domain matches the <code>defaultdomain</code> , then the entire substitution operation fails.
\$2I	Host domain (the canonical domain name for the domain the user is in), unless the host domain is the same as the defaultdomain option (in legacy configuration, the <code>service.defaultdomain</code> configutil parameter), or the MTA-specific override option ldap_default_domain ; when the host domain matches the <code>defaultdomain</code> , then no domain is inserted and also the leading character is removed. This substitution is typically used to remove a leading percent character, <code>%</code> , for such cases of the default domain.
\$3I	Host domain (the canonical domain name for the domain the user is in), unless the host domain is the same as the defaultdomain option (in legacy configuration, the <code>service.defaultdomain</code> configutil parameter), or the MTA-specific override option ldap_default_domain ; when the host

	domain matches the <code>defaultdomain</code> , then no domain is inserted and also the next character in the template is omitted. This substitution is typically used to remove a trailing slash character, <code>/</code> , for such cases of the default domain.
\$J	Host domain minus its first chunk
\$K	Filter constructed from the user and group object classes, as specified via the <code>ldap_user_object_classes</code> and <code>ldap_group_object_classes</code> MTA options
\$L	Username minus any special leading characters such as <code>~</code> or <code>_</code>
\$M	<code>uid</code> ; when the LDAP URL relates to a user address, this would be the actual <code>uid</code> value (or whatever LDAP attribute is named by the <code>ldap_uid</code> MTA option), but in other contexts may have another meaning (as for instance, <code>\$M</code> substitutes the detailed verdict string for spam/virus filter package integration <code>spamfilterN_action_M</code> MTA options)
\$nM	New in JES MS 6.2-0.01. Insert the <code>n</code> th character of the <code>uid</code> (or an "a" if the <code>uid</code> has no <code>n</code> th character)
\$N	Construct a comma-separated list of attributes (normally those to return in <code>reverse_url</code> lookups, hence a comma separated list of the attributes named by the <code>ldap_capture</code> , <code>ldap_recipientlimit</code> , <code>ldap_recipientcutoff</code> , <code>ldap_sourceblocklimit</code> , <code>ldap_preferred_language</code> , <code>ldap_personal_name</code> , <code>ldap_primary_address</code> , and <code>ldap_equivalence_addresses</code> MTA options, as well as (in JES MS 6.2-0.01 and later) <code>ldap_source_conversion_tag</code> , and (in JES MS 6.3-0.15 and later), <code>ldap_blocklimit</code> , <code>ldap_source_channel</code> , the <code>ldap_source_optin</code> MTA options, <code>ldap_preferred_country</code> , and the <code>ldap_spare_N</code> MTA options)
\$O	(New in JES MS 6.1p1/6.2) Substitute the source route
\$4O	(New in JES MS 6.1p1/6.2) Substitute the source route if present; if no source route is present, then instead remove the following character (normally used in a pattern to remove the colon character subsequent to the <code>---nonexistent---source route</code>)
\$P	Program name, that is, the value of the attribute named by the <code>ldap_program_info</code> MTA option (hence normally the value of the <code>mailProgramDeliveryInfo</code> attribute)
\$Q	Filter for address reversal in iMS 5.2p2; that is, a filter constructed from the attribute names listed in the <code>ldap_mail_reverses</code> MTA option (or the

	appropriate default attribute names, depending upon the schema tag, if <code>ldap_mail_reverses</code> is not set); this substitution sequence was typically used as the filter in <code>reverse_url</code> lookups in iMS 5.2, but with the Sun One Messaging Server 6.0 MTA, the <code>\$R</code> substitution sequence is normally used instead
<code>\$S</code>	Subaddress
<code>\$2S</code>	Subaddress if present, but if there is no subaddress, then remove the leading character (used to remove the plus character, +)
<code>\$R</code>	Filter for mail aliases; that is, a filter constructed from the attribute names listed in the <code>ldap_mail_aliases MTA option</code> (or with legacy configuration in the <code>local.imta.mailaliases</code> configutil parameter); as of JES MS 6.0 and later, also typically used as the search filter for <code>reverse_url</code> address reversal lookups
<code>\$T</code>	Turn domain name <code>chunk1.chunk2...chunkn</code> into <code>dc=chunk1,dc=chunk2,...,dc=chunkn</code>
<code>\$U</code>	The username portion of the address (that is, the local-part sans subaddress and sans any leading backwards-single-quote, tilde, or underscore character)
<code>\$nU</code>	Insert the <i>n</i> th character of the username
<code>\$V</code>	The base DN returned by calling <code>domainMap</code> on the domain in the address; note that this can be affected by the <code>domain_uplevel MTA option</code>
<code>\$1V</code>	A variant on the <code>\$V</code> substitution, where if the <code>\$V</code> <code>domainMap</code> sort of lookup fails (no <code>inetDomainBaseDn</code> or <code>aliasedObjectName</code> matching the domain name is found), then this <code>\$1V</code> substitution instead returns the DN of the top of the user and group tree, that is, the value of the <code>ugldapbasedn base option</code> (in legacy configuration, the <code>local.ugldapbasedn</code> configutil parameter) or as overridden via the MTA-specific <code>ldap_user_root MTA option</code>
<code>\$X</code>	<code>mailHost</code>
<code>\$nX</code>	Insert the <i>n</i> th component of the <code>mailHost</code>

Chapter 2 Options for migrating to Unified Configuration

2.1 <code>instancename</code> Option	2-1
2.2 <code>rolename</code> Option	2-1
2.3 <code>plugins</code> Option	2-1

There are three options in legacy configuration that provide meta-data for migrating from legacy configuration to Unified Configuration.

2.1 `instancename` Option

The `instancename` option specifies the name of this host's instance in the deployment. This option is used as a transition option when migrating to a Unified Configuration. Only used for purposes of migration.

2.2 `rolename` Option

The `rolename` option specifies the name of the role this host fulfills in the deployment (e.g., `backend-store`, `frontend-mta`, `relay`). This option is used as a transition option when migrating to a Unified Configuration. Only used for purposes of migration.

2.3 `plugins` Option

Enable notifications via ENS and/or JMQ by specifying a library name of `"libibiff"` or `"libjmqnotify"` respectively. Each library name should then be followed by an instance name preceded by a `"$"` character. Each library/instance pair should be separated from the next by a `"$"` character. Several instances of plugins may be specified with different instance names. The name given in the instance field for each specified plugin is the name used to look up the configuration for that plugin. The instance name `"ms-internal"` is reserved.

This option is not used for Unified Configuration.

Support for JMQ is deprecated and will be removed in a future release.



Chapter 3 Special symbolic names

A few special symbolic names may be used in option values in `msconfig`, as well as in [recipes](#).

```
IMTA_ROOT:      -> <serverroot> "/"
IMTA_LIB:       -> <serverroot> "/lib/"
IMTA_BIN:       -> <serverroot> "/lib"
IMTA_TABLE:     -> <configroot> "/"
IMTA_PROGRAM:  -> <dataroot> "/site-programs/"
IMTA_DL:        -> <dataroot> "/dl/"
IMTA_LOG:       -> <dataroot> "/log/"
IMTA_QUEUE:    -> <dataroot> "/queue/"
IMTA_HTTP:     -> <dataroot> "/www/"
IMTA_HOST:     -> local.hostname
IMTA_DEFAULTDOMAIN -> service.defaultdomain
IMTA_LIBUTIL   -> <serverroot> "/lib/libimtautil.so"
IMTA_LIBMAP    -> <serverroot> "/lib/libimtamap.so"
IMTA_TMP       -> tmpdir
IMTA_LANG      -> langdir
IMTA_VERIFY_RETURN -> return_verify
IMTA_RETURN_CLEANUP_PERIOD -> return_cleanup_period
IMTA_RETURN_SPLIT_PERIOD -> return_split_period
```

The value of the `SERVERROOT` environment variable is used to construct the `DATAROOT` value (`SERVERROOT/data`) and `CONFIGROOT` value (`SERVERROOT/config`).



Chapter 4 Recipe language

4.1 Comments	4-1
4.2 Integer values	4-1
4.3 Strings and list values	4-2
4.4 Variables	4-2
4.4.1 Optlists	4-3
4.4.2 Variable indices	4-3
4.5 Statements	4-4
4.6 User-defined routines	4-5
4.7 Operators	4-6
4.8 Functions	4-8
4.8.1 Configuration option access	4-13
4.8.2 File operations	4-14
4.8.3 Channel creation and manipulation operations	4-14
4.8.4 Mapping creation and manipulation operations	4-16
4.8.5 Optlist manipulation operations	4-17
4.9 Preprocessing Directives	4-18

Recipe files are used to automate configuration management tasks. Recipe files are expressed using a programming language. Since recipe files are intended to manipulate configuration information, many parts of which most naturally appear as names and string values or lists of strings, the recipe language is rather string oriented. The primary inspiration for the recipe language is the Icon programming language designed by Ralph Griswald. Recipe file syntax includes C-like expressions, operators, and assignments, Sieve-like conditionals, and loops. The available data types are integers, strings, and lists.

Recipes typically operate in three phases. First, a number of checks are done to make sure the right conditions exist for the recipe to be effective. Next the recipe asks a number of questions to determine exactly what changes should be made. Finally, the recipe implement the requested changes. Note that while this is the typical ordering, recipes are not constrained to use it and may use other approaches if appropriate.

4.1 Comments

The # character indicates that the remainder of a line is a comment.

4.2 Integer values

Integers are expressed as decimal values with an optional sign. An optional base specification prefix can be used to write values in other bases:

```
2%10 == 2
16%ff == 255
2%-1010 == -10
```

All integers are represented internally as signed 32 bit values.

4.3 Strings and list values

As recipe files are intended for manipulating configuration files and parameters, which are generally thought of most naturally as strings or lists of strings, the recipe language is rather string oriented.

A string value is written simply as characters within double quotes, *e.g.*,

```
"a sample string"  
"strings may  
include line breaks"
```

Backslashes have special meaning in string values:

```
"\" - quote  
"\t" - tab  
"\r" - carriage return  
"\n" - line feed  
"\" - backslash
```

A list value is written as a comma-separated list of elements, delimited by brackets, *e.g.*,

```
["e1", "e2", "e3", "e4"]
```

Note that the elements of a list are always strings.

4.4 Variables

Variables may have integer, string, or list values. Variables are created by assigning them a value; no type declarations are necessary. Variable names are case-insensitive.

```
v = 1;  
w = "this is a test";  
x = ["a", "b", "c"];
```

Variables may be used in most places where a string, integer, or list value is expected, including in list constructs. Given:

```
s = "string";  
l = ["list"];
```

the following expressions are true:

```
[s] == ["string"]  
[s,s] == ["string", "string"]  
[s,l] == ["string", "list"]  
[l,l] == ["list", "list"]
```

4.4.1 Optlists

Optlists are regular lists with an even number of elements. The elements are processed in pairs - the first element in a pair is the "name" element while the second element is the "value" element. A number of the built-in functions are designed to work with optlists. The special case of empty string as the first element in a pair is used to represent annotations - in this case the second element of the pair contains the annotation text.

4.4.2 Variable indices

Substrings and sublists may be referenced through the use of indices. Indices into strings and lists point not at characters, but between characters, as for strings in the Icon programming language. That is, 1 points just before the first character, 2 between the first and second character, *etc.*, and 0 points just after the last character, -1 points between the penultimate and last characters, -2 between the antepenultimate and penultimate characters, *etc.* The expression `s[i]` returns the character immediately following the interstice pointed to by the index.

For instance, if

```
s = "abcdef";
```

then the following expressions are true:

```
s[1] == "a"
s[3] == "c"
s[6] == "f"
s[-1] == "f"
s[-4] == "c"
s[-6] == "a"
```

whereas `s[0]` is illegal as it is trying to return the character after the last one in the string.

When a range is specified, then the substring between the two indices is returned. There is no question about whether this is inclusive or exclusive as the indices point at interstices. Thus again taking as a sample string:

```
s = "abcdef";
```

the following expressions are true:

```
s[1,0] == s
s[1,0] == "abcdef"
s[1,2] == "a"
s[2,0] == "bcdef"
s[2,-1] == "bcde"
s[-3, -1] == "de"
s[i,i] == "" # when 1 <= i <= length(s)+1
              # or -length(s) <= i <= 0
s[1,i+1] == left(s,i)
s[-i,0] == right(s)
```

Indices can be used on the left hand side of assignment statements. For example, after the assignment

```
s[1] = "z";
```

s will have the value "zbcdef".

List indices operate in a similar fashion, except that indices refer to list elements rather than characters. Single value list indices return a string while two-valued indices return a sublist. So if l is given a value

```
l = ["a", "b", "c", "d", "e", "f"];
```

the following expressions are all true:

```
l[1] == "a"
l[3] == "c"
l[6] == "f"
l[-1] == "f"
l[-4] == "c"
l[-6] == "a"
l[1,0] == ["a", "b", "c", "d", "e", "f"]
l[1,2] == ["a"]
l[2,0] == ["b", "c", "d", "e", "f"]
l[2,-1] == ["b", "c", "d", "e"]
l[3,-1] == ["c", "d", "e"]
l[-3,-1] == ["d", "e"]
```

4.5 Statements

The if...then...else... statements in the recipe language are akin to those of the [Sieve email filtering language](#):

```
if expression { ... }

if expression { ... }
else { ... }

if expression { ... }
elsif { ... }

if expression { ... }
elsif { ... }
else { ... }
```

A general loop construct is also provided, with the syntax:

```
loop {
    ...
    exitif (expression);
```

```
    ...
}
```

A loop may contain zero or more `exitif` statements. Loops may be nested:

```
loop {
    ...
    loop {
        ...
        exitif (expression-1); # Exit from inner loop #1
    }
    ...
    exitif (expression-2); # Exit from outer loop
    ...
    loop {
        ...
        exitif (expression-3); # Exit from inner loop #2
    }
}
```

Assignment statements are akin to those of the C programming language.

```
a = "string";
b = 2;
c = d = ["list"];
```

4.6 User-defined routines

As of the 8.0 release, the recipe language supports user-defined routines:

```
sub routine(p1, ...) {
    ...
    return expression-1;
}
...
variable = routine(expression-1, ...);
```

Up to 20 parameters are allowed. Parameters are passed by value and evaluation is lazy: An unused parameter is never evaluated. Parameters are only evaluated once, so it is easy to force evaluation to occur at the beginning of the routine:

```
sub f(x, y, z) {
    x; y; z;
    ...
}
```

The entire parameter list can be omitted if the routine requires no parameters.

Routines may call themselves recursively, e.g.,

```
sub factorial(n) {if n <= 1 {return 1;} else {return n * factorial(n-1);}}
```

Note, however, that since there is currently no mechanism for forward declarations of routines, two or more routines cannot call each other recursively.

Routines can reference and modify global variables. Local variables can also be defined by placing the `my` control command immediately prior to the first use of the variable:

```
sub fib(n) {
  my s = [1, 1];
  my a = 1;
  my b = 1;
  loop {
    exitif --n < 2;
    my c = a + b;
    s .= c;
    a = b;
    b = c;
  }
  return s;
}
```

Autoincrement, autodecrement, and the various augmented assignment operators (`+=`, `-=`, and so on) are all allowed on parameters and local variables. So is the exchange operator `:=`; however, exchange cannot be used with a global variable on the right hand side and a local variable or parameter on the left hand side.

4.7 Operators

The recipe language provides a variety of prefix, postfix, and infix operators. The following table lists the available operators in order of decreasing precedence.

Table 4.1 Operators in Order of Precedence

Operator	Description	Precedence
<code>?v</code>	Force interpretation of <code>v</code> as a variable, rather than as a pre-defined function	19
<code>f(...)</code>	Function call	18
Index		
<code>s1[i]</code>	Return the <i>i</i> th character of string/list <code>s1</code>	17
<code>s1[i, j]</code>	Return the <i>i</i> th through <i>j</i> th characters of string/list <code>s1</code>	17
Increment/decrement		
<code>v++</code>	Return <code>v</code> and then increment its value (<code>v</code> must be a variable with an integer value)	16
<code>v--</code>	Return <code>v</code> and then decrement its value (<code>v</code> must be a variable with an integer value)	16
<code>++v</code>	Increment the value of <code>v</code> and then return it (<code>v</code> must be a variable with an integer value)	16
<code>--v</code>	Decrement the value of <code>v</code> and then return it (<code>v</code> must be a variable with an integer value)	16
Unary bitwise and logical		

$\sim n$	Bitwise not	15
$!n$	Logical not	15
Arithmetic and concatenation		
$-n$	Integer negation	14
$+n$	Integer plus	14
$n * m$	Integer multiplication, string/list cross product	13
n / m	Integer division	13
$n \% m$	Integer modulus	13
$n + m$	Integer addition, string/list element by element concatenation	12
$n - m$	Integer subtraction	12
$n . m$	String/list concatenation	12
$n \ll m$	Left shift	11
$n \gg m$	Right shift	11
Comparisons		
$n < m$	Less than	10
$n \leq m$	Less than or equal to	10
$n \geq m$	Greater than or equal to	10
$n > m$	Greater than	10
$n == m$	Equal to	9
$n != m$	Not equal to	9
Infix bitwise		
$n \& m$	Bitwise and	8
$n \wedge m$	Bitwise xor	7
$n m$	Bitwise or	6
Infix logical		
$n \&\& m$	Logical and	5
$n \wedge\wedge m$	Logical xor	4
$n m$	Logical or	3
Conditional		
$p ? n : z$	n if p is nonzero, z otherwise	2
Assignment		
$v = e$	Assign v the value of e (v must be a variable)	1
$v += e$	Add/concatenate the value of e to v (v must be a variable)	1
$v -= e$	Subtract the value of e from v (v must be a variable with an integer value)	1
$v *= \text{exp}$	Multiply/cross product v by the value of e (v must be a variable)	1

<code>v /= e</code>	Divide <code>v</code> by the value of <code>e</code> (<code>v</code> must be a variable with an integer value)	1
<code>v &= e</code>	Bitwise and <code>e</code> into <code>v</code> (<code>v</code> must be a variable with an integer value)	1
<code>v = e</code>	Bitwise or <code>e</code> into <code>v</code> (<code>v</code> must be a variable with an integer value)	1
<code>v ^= e</code>	Bitwise xor <code>e</code> into <code>v</code> (<code>v</code> must be a variable with an integer value)	1
<code>v <<= e</code>	Left shift <code>v</code> by <code>e</code> (<code>v</code> must be a variable with an integer value)	1
<code>v >>= e</code>	Right shift <code>v</code> by <code>e</code> (<code>v</code> must be a variable with an integer value)	1
<code>v .= e</code>	Concatenate the value of <code>e</code> onto <code>v</code> (<code>v</code> must be a variable)	1
<code>v1 ::= v2</code>	Exchange the values in <code>v1</code> and <code>v2</code> (<code>v1</code> and <code>v2</code> must both be variables)	1

4.8 Functions

The recipe language provides a large number of built-in functions. The following table lists all of the built-in functions in alphabetical order; subsequent subsections describe all functions in groups.

Table 4.2 Alphabetical List of Built-in Functions

Function	Description
<code>add_channel(n, c)</code>	Adds a channel named <code>n</code> containing the optlist <code>c</code> . An error will be returned if the channel already exists. Each element of the optlist specifies a channel option and its corresponding value. An empty string must be specified for channel options that do not accept a value.
<code>add_group(g, c)</code>	Adds a group named <code>g</code> containing the optlist <code>c</code> . An error will be returned if the group already exists. Each element of the optlist specifies a group element and its corresponding value.
<code>abs(i)</code>	Return the absolute value of the numeric value <code>i</code> .
<code>add_mapping(n, c)</code>	Adds a mapping named <code>n</code> containing the optlist <code>c</code> . The mapping must not already exist. Each element of the optlist specifies either a mapping rule or an annotation. The name part of an optlist element specifies the rule pattern while the value part specifies the rule template.
<code>allof(i1[, i2...])</code>	Returns a nonzero (true) value if all of <code>i1</code> , <code>i2</code> , ... are nonzero; returns 0 otherwise.
<code>any(s1, s2)</code>	Return 2 if any character in <code>s1</code> appears as the first character of <code>s2</code> ; return 0 otherwise.
<code>anyof(i1[, i2...])</code>	Returns a nonzero (true) value if any of <code>i1</code> , <code>i2</code> , ... are nonzero, returns 0 if all are zero.
<code>append_mapping(n, c)</code>	Appends the contents of optlist <code>c</code> to the mapping named <code>n</code> . The mapping will be created if it doesn't already exist. Each element of the optlist specifies either a mapping rule or an annotation. The name part of an optlist element specifies the rule pattern while the value part specifies the rule template.

<code>append_group(n, c)</code>	Appends the contents of optlist <code>c</code> to the group named <code>n</code> . The group will be created if it doesn't already exist. Each element of the optlist specifies a group element and its corresponding value.
<code>append_rewrites(c)</code>	Appends the contents of optlist <code>c</code> to the current set of rewrite rules. Each element of the optlist specifies either a rewrite rule or an annotation. The name part of an optlist element specifies the rule pattern while the value part specifies the rule template.
<code>argc</code>	(New in 8.0) Returns the number of additional arguments given to the the <code>msconfig</code> run command.
<code>argv(n)</code>	(New in 8.0) Returns the <code>n</code> th argument given to the the <code>msconfig</code> run command as a string. The value <code>n</code> must be between 1 and <code>argc</code> inclusive.
<code>bal(c1, c2, c3, s)</code>	Scan <code>s</code> looking for an occurrence of a character in <code>c1</code> that is balanced with respect to <code>c2</code> and <code>c3</code> . Returns the position of the first balanced <code>c3</code> character in <code>s</code> if one is found, <code>length(s)+1</code> if no <code>c3</code> character is found but the string as a whole is balanced, or 0 if the string isn't balanced.
<code>chr(i1[, i2...])</code>	Returns a string containing successive characters with decimal values <code>i1, i2, ...</code>
<code>continue [s]</code>	The <code>continue</code> function does nothing if no warnings have been issued during the execution of the recipe. If one or more warnings have been issued by the recipe the administrator is prompted with the string <code>s</code> . An empty response or a response beginning with "Y" (yes) or "T" (true) will cause the script to continue running; any other response will cause the script to abort. A default prompt of "Some warnings have occurred. Do you want to continue [N]?" will be used if no value for <code>s</code> is specified.
<code>defined(s)</code>	Returns 1 if <code>s</code> is defined as a variable; return 0 otherwise.
<code>delete_channel(n)</code>	Delete the channel named <code>n</code> . No operation is performed if the channel does not exist.
<code>delete_group(n)</code>	Delete the group named <code>n</code> . No operation is performed if the group does not exist.
<code>delete_mapping(n)</code>	Delete the mapping named <code>n</code> . No operation is performed if the mapping does not exist.
<code>delete_options(l)</code>	Deletes all values associated with the options specified in the list <code>l</code> .
<code>delete_optlist(o, n...)</code>	Delete option <code>n</code> from optlist <code>o</code> and return the resulting modified list. The original optlist is returned if the specified option does not appear in the optlist. Additional option names can be specified to delete multiple options from the optlist.
<code>exists_channel(n)</code>	Returns a nonzero integer if the channel named <code>n</code> exists, zero if it does not.
<code>exists_file(s)</code>	Returns 1 if the file named by <code>s</code> exists, 0 if it doesn't.
<code>exists_mapping(n)</code>	Returns a nonzero integer if the mapping named <code>n</code> exists, zero if it does not.
<code>exists_option(n)</code>	Returns the number of values currently set for the option <code>n</code> . A value of 0 is returned if the option isn't set.
<code>exists_optlist(o, n)</code>	Returns 1 value if the the optlist <code>o</code> contains an option named <code>n</code> , 0 otherwise.

<code>find(s1,s2,i,j)</code>	Returns the position in <code>s2</code> of the first occurrence of <code>s1</code> in <code>s2[i:j]</code> .
<code>find(s,l,i,j)</code>	Returns the position in <code>l</code> of the first list element from <code>l[i:j]</code> that matches <code>s</code> .
<code>getenv(s)</code>	Return the value of the environment variable <code>s</code> . An empty string is returned if <code>s</code> is not defined. Note that <code>s</code> is case sensitive.
<code>get_channel(n)</code>	Returns the content of the channel named <code>n</code> as an optlist. An empty list is returned if the channel does not exist.
<code>get_mapping(n)</code>	Returns the content of the mapping named <code>n</code> as an optlist. An empty list is returned if the mapping does not exist.
<code>get_option(n)</code>	Returns the value of the option named <code>n</code> as a string. An error will occur if the specified option name is invalid or matches multiple options. An empty string is returned if the specified option is valid but not set.
<code>get_optlist(o,n)</code>	Returns the value of the option named <code>n</code> from the optlist <code>o</code> as a string. An empty string is returned if the specified option is not in the optlist.
<code>get_rewrites([p[,t]])</code>	Returns the selected set of rewrite rules as an optlist. <code>p</code> specifies a glob-style pattern to apply to the pattern part of each rewrite rule; only rules that match the pattern will be returned. Similarly, <code>t</code> specifies a glob-style pattern to apply to the template part of each rewrite rule; only the rules whose templates match the pattern will be returned. Both arguments are optional; if neither is specified then all rewrite rules are returned.
<code>get_path(r)</code>	(New in 8.0) Returns a path to a directory in the server instance. The argument <code>r</code> can be one of "server", "data", and "config", which will return the path to the server root, the data root, and configuration root directories, respectively.
<code>hash[:e][:h]v</code>	Returns the hash of the value <code>v</code> . <code>v</code> may be either a string or a list; if a list is specified a separate hash is computed for each element and returned as a new list. The hash <code>:h</code> may be any of <code>:md2</code> , <code>:md4</code> , <code>:md5</code> , <code>:sha1</code> , <code>:ripemd128</code> , or <code>:ripemd160</code> ; the default is <code>:sha1</code> . An encoding may optionally be applied; <code>:e</code> may be any of <code>:hexadecimal</code> , <code>:quotedprintable</code> , <code>:base64</code> , <code>:base85</code> , or <code>:binary</code> . <code>:binary</code> , or no encoding, is the default.
<code>hash_hmac[:e][:h]kv</code>	Returns the hmac of the value <code>v</code> using key <code>k</code> . <code>v</code> may be either a string or a list; if a list is specified a separate hmac is computed for each element and returned as a new list. The underlying hash function <code>:h</code> may be any of <code>:md2</code> , <code>:md4</code> , <code>:md5</code> , <code>:sha1</code> , <code>:ripemd128</code> , or <code>:ripemd160</code> ; the default is <code>:sha1</code> . An encoding may optionally be applied; <code>:e</code> may be any of <code>:hexadecimal</code> , <code>:quotedprintable</code> , <code>:base64</code> , <code>:base85</code> , or <code>:binary</code> . <code>:binary</code> , or no encoding, is the default.
<code>integer(e)</code>	Converts <code>e</code> to an integer. If <code>e</code> is already an integer it is returned unchanged; if <code>e</code> is a string it is read as a sequence of ASCII digits. If <code>e</code> is a list it must contain one element and that element is treated in the same way a string would be.
<code>lcase(e)</code>	Converts any upper case characters in <code>e</code> to lower case. If <code>e</code> is a number it is converted to a string.
<code>ldap_init(l)</code>	Initializes the built in LDAP client. This call must be performed prior to using any of the other <code>ldap_*</code> functions. The client uses the current settings of the configuration options <code>ugldaphost</code> , <code>ugldapbinddn</code> , <code>ugldapbindcred</code> , <code>ugldapport</code> , and <code>ugldapusessl</code> when it

	initializes. The single argument <i>l</i> is an optlist specifying override values for any or all of these options. The optlist may be empty. Repeated calls will shut down and reinitialize the LDAP client with new settings. The LDAP client is shut down automatically when the recipe terminates. New in 8.0.1.
<code>ldap_ldif(l[, f])</code>	Apply the LDIF file specified in the string <i>l</i> to the LDAP directory. Any LDAP error that occurs will be treated as a recipe warning (but see the bit 12 in the flag argument). The optional argument <i>f</i> is a bit-encoded integer specifying a number of flags. The currently defined flag bits are: Bit 0 (value 1) - if set, continue processing after any error, Bit 1 (value 2) - if set, treat "entry exists" on add as success, Bit 2 (value 4) - if set, allow no-such-object if hint present, and Bit 12 (value 4095) - if set, treat any LDAP error that occurs as a recipe error. An integer count of the number of successful modifications performed is returned. <code>ldap_init</code> must be called before calling <code>ldap_ldif</code> . New in 8.0.1.
<code>left(s1, i[, s2])</code>	Returns the leftmost <i>i</i> characters of <i>s1</i> . If <i>i</i> is greater than <code>length(s1)</code> the result is padded with <i>s2</i> . As much of <i>s2</i> as is necessary will be used; if <i>s2</i> is too short it will be used multiple times. <i>s2</i> defaults to a space if it is omitted.
<code>left(l1, i[, l2])</code>	Returns the leftmost <i>i</i> elements of <i>l1</i> . If <i>i</i> is greater than <code>length(l1)</code> the result is padded with <i>l2</i> . As much of <i>l2</i> as is necessary will be used; if <i>l2</i> is too short it will be used multiple times. <i>l2</i> defaults to one empty list element if it is omitted.
<code>length(s)</code>	Returns the number of 8-bit characters in the string <i>s</i> .
<code>length(l)</code>	Returns the number of elements in the list <i>l</i> .
<code>list(s, n)</code>	Returns a list <i>n</i> elements long with each element equal to <i>s</i> .
<code>list(l, n)</code>	Returns a list consisting of <i>n</i> copies of <i>l</i> .
<code>make_path(p)</code>	Converts a path <i>p</i> using <code>IMTA_TABLE:</code> and similar MTA-specific constructs into a proper file path. Note that the path that's constructed may only be valid on the system where <code>msconfig</code> is running.
<code>match(r, s)</code>	Returns 1 (true) if the regular expression <i>r</i> matches a substring of string <i>s</i> , 0 (false) otherwise. Note that the pattern <i>r</i> may be prefixed with "^" (match beginning of line) and suffixed with "\$" (match end of line) to require a full string match. The regular expression vocabulary is compatible with that of the TCL/TK scripting language.
<code>max(i, j[, ...])</code>	Returns the largest element in a set of integers.
<code>max(s1, s2[, ...])</code>	Returns the largest element in a set of strings.
<code>min(i, j[, ...])</code>	Returns the smallest element in a set of integers.
<code>min(s1, s2[, ...])</code>	Returns the smallest element in a set of strings.
<code>pop(l)</code>	Returns the first element of list <i>l</i> . The element is deleted from the list. <i>l</i> must be a variable with a list value.
<code>prepend_mapping(n, c)</code>	Prepends the contents of optlist <i>c</i> to the mapping named <i>n</i> . The mapping will be created if it doesn't already exist.
<code>print s</code>	Print the string <i>s</i> on the administrator's terminal.
<code>push(l, s)</code>	Adds the string <i>s</i> to the beginning of list <i>l</i> . The list <i>l</i> is updated as well as being returned. <i>l</i> must be a variable with a list value.

<code>read_file(s)</code>	Reads and returns the content of the file named by <code>s</code> . An error occurs if the file doesn't exist or cannot be read. Line feeds are used as line separators.
<code>repl(s, j)</code>	Returns a string consisting of <code>j</code> concatenations of <code>s</code> .
<code>repl(l, j)</code>	Returns a list consisting of <code>j</code> concatenations of <code>l</code> .
<code>replace_channel(n, c)</code>	Replaces the channel named <code>n</code> with the contents of optlist <code>c</code> . The channel will be created if it doesn't already exist.
<code>replace_mapping(n, c)</code>	Replaces the contents of the mapping named <code>n</code> with the contents of optlist <code>c</code> . The mapping will be created if it doesn't already exist.
<code>reverse(s)</code>	Reverses all the characters in <code>s</code> and returns the result.
<code>reverse(l)</code>	Reverses all the elements in <code>l</code> and returns the result.
<code>right(s1, i[, s2])</code>	Returns rightmost <code>i</code> characters of <code>s1</code> . If <code>i</code> is greater than <code>length(s1)</code> the result is padded with <code>s2</code> . As much of <code>s2</code> as is necessary will be used; if <code>s2</code> is too short it will be used multiple times. <code>s2</code> defaults to a space if it is omitted.
<code>right(l1, i[, l2])</code>	Returns rightmost <code>i</code> elements of <code>l1</code> . If <code>i</code> is greater than <code>length(l1)</code> the result is padded with <code>l2</code> . As much of <code>l2</code> as is necessary will be used; if <code>l2</code> is too short it will be used multiple times. <code>l2</code> defaults to one empty list element if it is omitted.
<code>put_optlist(o, n, v...)</code>	Put the option <code>n</code> with the value <code>v</code> on optlist <code>o</code> and return the resulting modified list. The option's value is replaced if the option is already present. Additional name-value pairs can be specified in the call to <code>put</code> multiple options on the optlist.
<code>set_channel(n, c)</code>	Modify the channel named by <code>n</code> with the channel options specified by the optlist <code>c</code> . The channel will be created if it doesn't already exist.
<code>sign(i)</code>	Returns -1 if <code>i < 0</code> , 0 if <code>i = 0</code> , +1 if <code>i > 0</code> .
<code>set_option(n, v)</code>	Set option <code>n</code> to the string value <code>v</code> . An error will occur if the specified option name is invalid or matches multiple options.
<code>set_options(o)</code>	Set zero or more options specified in optlist form. Option names and values are taken from optlist <code>o</code> in the obvious way. An error will occur if any of the specified option names are invalid or match multiple options.
<code>sort(l1[, i[, l2]])</code>	Sorts the elements of <code>l1</code> to be in ascending order if <code>i < 0</code> and descending order if <code>i = 0</code> . <code>i</code> defaults to 1 if it is omitted. If <code>l2</code> is present its elements are shifted in the same way as elements in <code>l1</code> are shifted.
<code>split(s[, c[, i]])</code>	Produces a list of elements consisting of pieces of <code>s</code> delineated by characters in <code>c</code> . If omitted <code>c</code> defaults to a comma. If <code>i</code> is 0 or 1, zero length elements are preserved; if <code>i</code> is 2, they are not. If omitted <code>i</code> defaults to 1.
<code>split(l[, c[, i]])</code>	Produces a list of elements consisting of pieces of elements of <code>l</code> delineated by characters in <code>c</code> . If omitted <code>c</code> defaults to a comma. If <code>i</code> is 0, boundaries between the original elements aren't preserved and zero length elements can be output; if <code>i</code> is 1, boundaries are preserved and zero length elements can be output; if <code>i</code> is 2, boundaries aren't preserved and zero length elements are omitted. If omitted <code>i</code> defaults to 1.

<code>string(e)</code>	Converts <code>e</code> to a string. If <code>e</code> is already a string it is returned unchanged. If <code>e</code> is an integer it is converted to a string. If <code>e</code> is a list, the string that results from concatenating the elements of <code>e</code> is returned.
<code>translate(s1,s2,s3)</code>	Interprets the string <code>s1</code> as being in the character set specified by <code>s2</code> and returns a version translated into the character set specified by <code>s3</code> .
<code>trim(s[,c])</code>	Returns <code>s</code> with any trailing characters found in <code>c</code> removed. <code>c</code> defaults to space and tab if omitted.
<code>trim(l[,c])</code>	Returns <code>l</code> with any trailing characters found in <code>c</code> removed from each element. <code>c</code> defaults to space and tab if omitted.
<code>type(e)</code>	Returns "integer" if <code>e</code> evaluates to an integer, "string" if <code>e</code> evaluates to a string, and "list" if <code>e</code> evaluates to a list.
<code>ucase(e)</code>	Converts any lower case characters in <code>e</code> to upper case. If <code>e</code> is a number it is converted to a string.
<code>unset_option(n)</code>	Removes option <code>n</code> from the configuration. An error will occur if the specified option name is invalid or matches multiple options.
<code>write_file(s1,s2)</code>	Writes the contents of string <code>s2</code> into the file named by <code>s1</code> . An error occurs if the file cannot be opened or written. Line feeds should be used as line separators in the string.
<code>write_file(s1,l[,s2])</code>	Writes the contents of the <code>l</code> into the file named by <code>s1</code> . Each list element written is terminated by the value of <code>s2</code> ; line feed is the default terminator if <code>s2</code> isn't supplied. An error occurs if the file cannot be opened or written.
<code>write_optlist(o)</code>	Convert the optlist <code>o</code> to a series of "name=value" lines and return the resulting string containing those lines. The string is in a format suitable for writing to a file with <code>write_file</code> .

4.8.1 Configuration option access

The primary purpose of the recipe language is to manipulate the various option settings in a Messaging Server configuration. This functionality is provided by a number of separate functions. These functions all accept either an option name or optlist containing option name/value pairs as arguments. Some functions allow incomplete option names and/or wildcards while others do not.

The existence of an option setting can be determined with the `exists_option` function. This function accepts an option name string as an argument and returns a count of the number of options set in the configuration that match the name. 0 (false) is returned if no options are set that match the name. For example:

```
exists_option("os_debug")      -> 0  (os_debug is not currently set)
exists_option("channel:tcp_*") -> 42 (42 options are set on tcp_channels)
```

The `get_option` function returns the value of a single option. An error will occur if the name given matches multiple options or does not exist. An empty string will be returned if the specified option is valid but is not set.

```
get_option("mm_debug")        -> 0      (mm_debug is set to 0)
get_option("os_debug")        -> ""     (os_debug is not set)
```

```
get_option("*_debug")          -> <error> (multiple *_debug options)
```

4.8.2 File operations

Although recipes primarily operate directly on Messaging Server configuration data without any need for explicit file operations, situations may arise where additional files need to be read or written. Accordingly, a set of file manipulation functions is provided in the recipe language.

If no explicit path is given in the file name the file location defaults to the config root directory. Explicit paths may be specified or any of the following special prefixes may be used:

```
IMTA_ROOT:      -> <serverroot> "/"
IMTA_LIB:       -> <serverroot> "/lib/"
IMTA_TABLE:     -> <configroot> "/"
IMTA_PROGRAM:  -> <dataroot> "/site-programs/"
IMTA_LOG:       -> <dataroot> "/log/"
IMTA_QUEUE:    -> <dataroot> "/queue/"
IMTA_HTTP:     -> <dataroot> "/www/"
```

The `exists_file(s)` function returns 1 if the file named by `s` exists, 0 if it doesn't.

Files may be read with `read_file(s)`. The contents of the file named by `s` are returned as a string. Line feeds are used as line delimiters in the string. For example:

```
split(trim(read_file("a.a")), "\n"), "\n")
```

returns the contents of the file with each line as a list element and any trailing blank lines removed.

There are two ways to write files. `write_file(s1, s2)` writes the contents of string `s2` to the file named by string `s1`. `write_file(s1, l[, s2])` writes the contents of the `l` into the file named by `s1`. Each list element written is terminated by the value of `s2`; line feed is the default terminator if `s2` isn't supplied. In either form an error occurs if the file cannot be opened or written.

4.8.3 Channel creation and manipulation operations

An MTA channel consists of a named set of option-value pairs, usually containing an `official_host_name` option. A single channel is represented in the recipe language using an `optlist`, and a number of functions are provided to access and manipulate channels. All of these functions accept the name of the channel as the first argument. This name may be specified in any case and is converted to lower case.

Channel existence can be checked with `exists_channel`. A nonzero value is returned if the named channel is already part of the configuration; zero if it isn't.

The contents of a mapping can be retrieved as an `optlist` using the `get_channel` function. For example, if the configuration has a `tcp_tas` channel:

```
tcp_tas deliveryflags 2 mustsaslsrvr smtp allowswitchchannel maytlssrvr
```

tcp_tas-daemon

The call `get_channel("tcp_tas")` will return an optlist:

```
[ "official_host_name", "tcp_tas-daemon",
  "deliveryflags", "2",
  "mustsaslsrver", "",
  "smtp", "",
  "allowswitchchannel", "",
  "maytlssrver", "" ]
```

Note that channel options that do not accept a value appear with a zero length string as the value.

The `add_channel` function is used to add a new mapping to the configuration. A second argument is required specifying the various channel options as an optlist. An error is returned if the channel already exists. For example, the call:

```
add_channel("tcp_aol", [ "official_host_name", "tcp-aol",
                        "single_sys", "",
                        "randommx", "",
                        "noswitchchannel", "",
                        "pool", "SMTP_POOL",
                        "smtp", "" ]);
```

adds this channel to the configuration:

```
tcp_aol single_sys randommx noswitchchannel pool SMTP_POOL smtp
tcp-aol
```

The `replace_channel` function is the same as `add_channel`, except that any channel with that name that already exists will be removed prior to the addition.

The `delete_channel` function deletes the named channel from the configuration. No operation is performed if the channel doesn't exist.

Finally, the `set_channel` change an existing channel. The second argument to must be an optlist containing the channel options to set. Existing options will be overridden; new options will be added. For example, given the channel:

```
tcp_intranet loopcheck maysaslsrver mx pool SMTP_POOL \
  saslsrver switchchannel tcp_auth single_sys smtp \
  allowswitchchannel maytlssrver
tcp_intranet-daemon
```

The call:

```
set_channel("tcp_intranet", [ "master_debug", "",
                              "nomx", "",
                              "daemon", "router.example.com",
                              "multiple", "" ]);
```

Will modify the channel to be:

```
tcp_intranet daempn router.example.com loopcheck master_debug \  
  maysaslserver multiple nomx pool SMTP_POOL \  
  saslswitchchannel tcp_auth single_sys smtp \  
  allowswitchchannel maytlserver  
tcp_intranet-daemon
```

4.8.4 Mapping creation and manipulation operations

An MTA mapping consists of a named and possibly annotated set of pattern-template pairs. A single mapping is represented in the recipe language using an optlist, and a number of functions are provided to access and manipulate mappings. All of these functions accept the name of a mapping as the first argument. This name may be specified in any case and is converted to upper case.

Mapping existence can be checked with `exists_mapping`. A nonzero value is returned if the named mapping is already part of the configuration; zero if it isn't.

The contents of a mapping can be retrieved as an optlist using the `get_mapping` function. For example, if the configuration has a `PORT_ACCESS` mapping:

```
PORT_ACCESS  
  
! Handle internal IP address  
  *|*|*|*|*          $$ | INTERNAL_IP; $3 | $$E  
  *                  $NEXTERNAL
```

The call `get_mapping("PORT_ACCESS")` will return an optlist:

```
[ "", " Handle internal IP addresses\n",  
  "*|*|*|*|*", "$$ | INTERNAL_IP; $3 | $$E",  
  "**", "$NEXTERNAL" ]
```

Note that the comment appears as name-value pair with an empty string as the name.

The `add_mapping` function is used to add a new mapping to the configuration. A second argument is required specifying the content of the mapping as an optlist. An error is returned if the mapping already exists. For example, the call:

```
add_mapping("test_mapping", [ "a", "b", "c", "d", "", "Last", "e", "f" ] );
```

adds this mapping to the configuration:

```
TEST_MAPPING  
  
  a b  
  c d  
! Last  
  e f
```

The `replace_mapping` function is the same as `add_mapping`, except that if the mapping already exists its contents will be replaced.

The `delete_mapping` function deletes the named mapping from the configuration. No operation is performed if the mapping doesn't already exist.

Finally, the `append_mapping` and `prepend_mapping` functions add entries to an existing mapping. The second argument to these functions must be an optlist containing the entries to add. Both functions are equivalent to `add_mapping` if the specified mapping doesn't already exist. For example, given the mapping:

```
TEST_MAPPING
```

```
  c d
```

The calls:

```
prepend_mapping("Test_Mapping", ["a","b"]);
append_mapping("test_mapping", ["e","f"]);
```

Will modify the mapping to be:

```
TEST_MAPPING
```

```
  a b
  c d
  e f
```

4.8.5 Optlist manipulation operations

As previously described, an optlist is a list containing an even number of strings which are interpreted as name-value pairs. A number of functions are provided to manipulate these sorts of lists.

An optlist can be created and populated just like any other list. Alternately, the `put_optlist` function can be used to add elements to an empty optlist. Optlists can also be read from files in name=value format. For example:

```
o = []; # Empty optlist
o = ["A","B"]; # Optlist containing a single option A with value B
o = ["A","C", "B", "D"]; Optlist containing two options A and B with values C and D
o = put_optlist([], "A","C", "B", "D"); # Same as previous optlist
o = read_optlist(read_file("optlist.txt")); # Read optlist from file optlist.txt
```

You can get option values from an optlist or check if a given option exists. For example, given an optlist `o = ["A", "C", "B", "D"]`, the following results would be returned:

```
get_optlist(o, "A") -> "C"
get_optlist(o, "B") -> "D"
get_optlist(o, "E") -> ""
```

```
exists_optlist(o, "A") -> 1
exists_optlist(o, "N") -> 0
```

Options can be set or deleted from an optlist. Note that it is common to assign the results of these functions back to the same optlist.

```
o = put_optlist(o, "E", "F"); # Add option E with value F to optlist o
o = delete_optlist(o, "A"); # Delete option A from optlist o
```

Optlists can be written out as name=value format files:

```
write_file("optlist.txt", write_optlist(o));
```

4.9 Preprocessing Directives

The recipe language provides a very limited processing facility as of the 8.0.1 release. The following preprocessing directives are supported:

Table 4.3 Preprocessor Directives

Directive	Description
<code>%define <i>name</i> [<i>value</i>]</code>	Define the preprocessor symbol <i>name</i> . A optional value can be specified but nothing presently makes use of such values.
<code>%elifdef <i>name</i></code>	Combines <code>%code</code> and <code>%ifdef</code> .
<code>%elifndef <i>name</i></code>	Combines <code>%code</code> and <code>%ifndef</code> .
<code>%else</code>	Invert the processing state of the preceding <code>%ifdef</code> or <code>%ifndef</code>
<code>%endif</code>	Terminates the innermost <code>%ifdef</code> or <code>%ifndef</code> processing block. The processing state reverts to that of the enclosing block if there is one.
<code>%ifdef <i>name</i></code>	Only process subsequent material up to a matching <code>%else</code> or <code>%endif</code> if <i>name</i> is defined.
<code>%ifndef <i>name</i></code>	Only process subsequent material up to a matching <code>%else</code> or <code>%endif</code> if <i>name</i> is not defined.
<code>%include <i>filename</i></code>	Include the content of the file <i>filename</i> in the recipe at the point where the <code>%include</code> directive appears.
<code>%undef <i>name</i></code>	Remove a previous definition of <i>name</i> .

Preprocessing directives must appear in column 1 to be recognized. Note that preprocessor directives have lower precedence than quoted strings, so directives won't be recognized inside of multiline quoted strings.

Chapter 5 Sieve filters

5.1 Sieve language	5-2
5.1.1 Brief overview of Sieve language elements	5-3
5.1.2 Sieve supported extensions	5-16
5.2 Sieve hierarchy	5-65
5.2.1 Sieve filters: types of scripts	5-65
5.2.2 Sieve filters: semantics of multiple scripts	5-67
5.2.3 Sieve filters: evaluation of multiple scripts	5-67
5.3 Sieve filters: implementation internals	5-72
5.4 Head of household Sieve filters	5-73

[RFC 5228 \(Sieve\)](#) (originally [RFC 3028 \(Sieve\)](#)), later updated by [various extensions](#) and [proposed extensions](#), defined a [language](#) for specifying processing of messages appropriate for performing upon message delivery. Such processing might include: filing those messages meeting specified criteria into special folders rather than simply delivering into the INBOX, redirecting (so-called "forwarding") messages meeting specified criteria to additional recipients, setting IMAP flags for messages meeting specified criteria, generating new notification messages when certain sorts of messages are delivered, returning "vacation" messages, discarding messages matching specified criteria, *etc.*

The MTA supports a [hierarchy of Sieve filters](#) applicable to messages. At the user level and domain level, this includes [user personal Sieve filters](#), so-called "head of household" or "parental" Sieve filters, and [domain level \(domain wide\) Sieve filters](#). And certain user LDAP attributes are interpreted by the MTA as specifying a Sieve "vacation" action---so essentially converted on-the-fly into a Sieve pseudo-script---and then merged with whatever other, explicit, user Sieve actions apply. The MTA also supports system (MTA) level Sieve filters, including [channel level Sieve filters \(for either or both of destination channels and source channels\)](#) and an [MTA-wide Sieve filter](#).

The MTA's [spam/virus filter package integration](#) also takes the form of Sieve filter scriptlets, where the MTA is configured to interpret possible [spam/virus filter package verdicts](#) as [requests to apply specified Sieve filter actions](#).

An important (and rather complex) topic is the interaction and hierarchy of how the MTA merges these multiple "levels" of Sieve scripts (and pseudo-scripts), and which Sieve filter actions take precedence, when multiple levels of Sieve script apply to a message. See the [Sieve hierarchy](#) topic for further discussion.

The MTA evaluates and applies applicable Sieve scripts (per its [Sieve hierarchy](#) rules) during message enqueue processing. While system-level Sieve scripts are often applicable (depending upon type and configuration) early in a message's lifetime, such as upon initial message submission to an MTA, user-level Sieve scripts for local recipients tend to be applicable instead at the time of enqueue to a delivery channel (enqueue to an [ims-ms channel](#) or [tcp_lmtpcs* channel](#)).

A separate and different use of Sieve filters is available to the Message Store, for message expiration purposes. If enabled with the [expiresieve](#) Message Store option, the Message Store can use Sieve filter tests to determine which messages to expire.

5.1 Sieve language

The Sieve filter language was originally defined in [RFC 3028 \(Sieve: A Mail Filtering Language\)](#), since updated by [RFC 5228 \(Sieve: An Email Filtering Language\)](#). The Sieve language provides a way to analyze Internet format messages ([RFC 822](#) messages), and perform processing appropriate for performing upon message delivery. Such processing might include: filing those messages meeting specified criteria into special folders rather than simply delivering into the INBOX, redirecting (so-called "forwarding") messages meeting specified criteria to additional recipients, setting IMAP flags for messages meeting specified criteria, generating new notification messages when certain sorts of messages are delivered, returning "vacation" messages, discarding messages matching specified criteria, *etc.* Additional RFCs, proposed extensions, and MTA-private extensions, have further extended and modified the Sieve language; see [Sieve supported extensions](#) for a list.

The RFCs defining standard Sieve features, and the Internet drafts for proposed Sieve extensions, are the most definitive resource for undering Sieve language syntax. For RFCs, see

<http://tools.ietf.org/rfc/>

and for Internet drafts, search in the "Individual Submissions" area at

<http://tools.ietf.org/id/>

Here follows a very brief overview of Sieve.

A Sieve script consists of a sequence of commands. Commands are [tests](#), [actions](#), or [control structures](#). (There are some special cases in the MTA's Sieve implementation, as for instance the MTA allows "size" to be used not only in its standard capacity as a test, but also as a function call. And new in MS 8.0, the MTA supports private operators "memcache" and "metermaid" which have uses both as actions and as tests.) Many actions and tests may take arguments, both positional and tagged, or have modifiers.

The values in Sieve scripts are generally strings or non-negative integers. However, values are also subject to a few alternate forms; see in particular the [variables](#) and [encoded-character](#) extensions. And the MTA's Sieve implementation supports use of signed integers (and in particular, negative integers). Furthermore, the MTA's Sieve implementation supports the use of expressions in places where the base Sieve specification expects values.

Many Sieve extensions, both standard and at proposal stage, plus additional extensions private-to-the-MTA, are supported by the MTA, adding various additional actions, commands, and control structures to the base Sieve language. Extensions, especially standardized extensions, generally need to be enabled using a "require" control structure. An attempt to use an invalid action, test, or control structure will result in an "Undefined function or variable "name" referenced" error. An attempt to "require" an unsupported or unenabled Sieve extension will result in an "Unknown function required: name" error. (Such errors are reported in an email message to the Sieve "owner" -- the user to whom the Sieve belongs in the case of user-level Sieves, or the [postmaster](#) in the case of system-level Sieves.)

Note that in addition to supporting private extensions, more generally the MTA also supports an extended Sieve syntax, including allowing expressions where Sieve expects arguments, and allowing assignment statements. The MTA also supports extending the Sieve language via [custom tests](#) defined via MTA mapping tables. And because the MTA's Sieve implementation

is built on top of the MTA's implementation of string and mathematical operation processing, the MTA's Sieve implementation supports some string functions and mathematical functions (and supports additional numeric forms, such as negative integers) not part of the base Sieve specification.

Note that as the Sieve filter language has been undergoing rather rapid development, support for additional language elements will likely be added in future. See release notes for current versions of the MTA software for notices of additional language element support.

5.1.1 Brief overview of Sieve language elements

For convenience, [Sieve language elements](#) provides a tabular overview of Sieve language elements. [Table of Sieve language elements](#) summarizes the basic Sieve language elements (but not the subelements that are standard under an element), plus any supported extension elements and extension subelements. For full descriptions of Sieve language elements, see the referenced RFCs and drafts. In [Table of Sieve language elements](#), arguments/values are shown in italics, optional elements are enclosed in square brackets ([]), choices are enclosed in angle brackets (< >) with the distinct choices separated by the forward slash character (/), default choices are shown in bold type, and optional repetition of an element is indicated with the asterisk character (*).

Table 5.1 Sieve language elements

Element syntax				
Modifier	Source	Restrictions	Main capability	Description
Modifier			Additional capability	
Control structures				
{...}	RFC 3028			Block of commands
<i>error string</i>	RFC 5463		require "ihave" ;	Terminate Sieve script with runtime error
foreverypart [<i>:name string</i>] <i>command-block</i>				
	RFC 5703		require "foreverypart" ;	(New in MS 8.0) Loop through the MIME parts of a message
<i>:name</i>	RFC 5703			Specify a name for this <i>foreverypart</i> loop for reference in enclosed <i>break</i> or <i>continue</i> statements
break [<i>:name string</i>]	RFC 5703			(New in MS 8.0) Break out of a <i>foreverypart</i> loop
<i>:name</i>	RFC 5703			Terminate closest enclosing loop having specified name
continue [<i>:name string</i>]	Private			(New in MS 8.0) Pass control to the bottom of the <i>foreverypart</i> loop
<i>:name</i>	RFC 5703			(New in MS 8.0) Pass control to the bottom of the closest enclosing <i>foreverypart</i> loop having specified name
loop [<i>exitif expression</i>]* <i>command-block</i>				
	Private	system-level		General loop
exitif	Private	system-level		Exit a <i>loop</i> structure
if <i>test command-block</i> [elsif <i>test command-block</i>]* [else <i>command-block</i>]				
	RFC 3028			Branch-on-condition control structure
elsif <i>test command-block</i>	RFC 3028			Next case of branch-on-condition structure
else <i>test command-block</i>	RFC 3028			Final case of branch-on-condition structure
require <i>capability-list</i>	RFC 3028	strict_require MTA option		Declare that a Sieve script may use the named extension(s)
stop	RFC 3028			End processing
Actions				
addconversiontag <i>string-or-list</i>				
	Private	system-level		System-level Sieve action to add the specified conversion tag(s) to the message
addflag [<i>variable-name</i>] <i>list-of-flags</i>				
	RFC 5232	(max_variables MTA option, if using optional variable argument)	require "imap4flags" ; (or require	Add the specified IMAP flag(s) to the message

Brief overview of Sieve language elements

			["imap4flags", "variables"]; if using optional variable argument)	
addheader [:last] header-field-name value-string				
	RFC 5293	max_addheaders MTA option		Add the specified header line (by default, at the beginning of the existing message header)
:last	RFC 5293			Add the specified header line at the end of the existing message header
:replace	Private			Add the specified header line, removing any previously present such header line(s)
addprefix string				
	Private			Add prefix text to the beginning of the first plain text part of the message
addsufffix string				
	Private			Add suffix text to the end of the first plain text part of the message
addtag string				
	Private			Add a tag (prefix text) to the Subject: header line
adjustcounter [:channel channel-name] [:duplicate] counter-name [value]				
	Private	system-level		(New in MS 8.0) System-level Sieve action to adjust value of a system Sieve accessible MTA counter
:channel	Private	system-level		Name of channel, one of whose counters is to be adjusted; the current source channel is assumed if no channel is explicitly specified
:duplicate	Private	system-level		By default, "adjustcounter" is suppressed if the Sieve script is reevaluated (as when a Sieve script that uses an envelope "To" test applies to a message addressed to multiple recipients); specifying the ":duplicate" modifier means that the "adjustcounter" action will be performed even upon reevaluation
capture [:dsn / :message / :journal] [:header] repository-address				
	Private	system-level+		Capture a message copy for legal intercept, archival, message replay, or similar purposes
:dsn	Private	system-level		Generate an encapsulated (DSN format) capture message; ":dsn" is the default
:header	Private	system-level		Capture message contains only the header of the original message, not the body; cannot be combined with ":message"
:journal	Private	system-level		Generate a MS Exchange "journal" format capture message
:message	Private	system-level		Generate a capture message as a pure, unencapsulated message
debug list-or-string				
	Private	mm_debug MTA option		Output the specified debug string; when mm_debug is 2 or more, the string is output to a debug log file, or with <code>imsimta test -expression -mm -debug=2</code> the specified string is output to the terminal
deleteheader [:index value [:last]] [MATCH-TYPE] [COMPARATOR] header-field-name-string [value-patterns-list]				
	RFC 5293		require "edithheader";	Delete the specified header line; (note that per RFC 5293, attempts to delete Received: or Auto-submitted: header lines will be ignored)
:index	RFC 5293			Separate though equivalent to the RFC 5260 ":index" MATCH-TYPE: attempt to match (and hence potentially delete) only exactly the specified occurrence of the specified header line, by default starting counting from the beginning of the message header
:last	RFC 5293			For the ":index" modifier, count backwards from the end of the message header
discard [log-string]				
	RFC 3028, plus private logging argument	filter_discard and log_filter MTA options, logging channel option		Discard message (do not deliver, just delete); a log-string may be specified to be included in the log_filter field in MTA message transaction log entries
ereject reason-string				
	RFC 5429	enable_sieve_ereject MTA option	require "ereject";	Refuse message during SMTP transaction (or generate DSN if SMTP level rejection is not possible); note that use of non-US-ASCII characters in the reason-string prevents the SMTP level rejection
extracttext [MODIFIER] [:first number] variable-name				
	RFC 5703	max_variables MTA option	require ["extracttext",	(New in MS 8.0) Store extracted message text into a variable

Brief overview of Sieve language elements

			"foreverypart", "variables";	
:first	RFC 5703			Extract only the first N characters of the current MIME body part into a variable
:encodeurl	RFC 5435			In a <code>variablesset</code> command or an <code>extracttext</code> command, perform percent-encoding (as per RFC 3986) of the string value
:length	RFC 5229			Decimal number of characters in the extracted string, converted to a string
:lower	RFC 5229			Convert upper case characters to lower case
:lowerfirst	RFC 5229			Convert first character to lower case if it is a letter and upper case; rest of extracted string left unchanged
:quoteregex	Private			In a <code>variablesset</code> command or <code>extracttext</code> command, backslash quote any characters requiring quoting for regex, namely asterisk, question mark, or backslash (*, ?, \)
:quotewild	Private			Prefix "*", "?", "\" characters with "\"
:quotewildcard	RFC 5229			Prefix "*", "?", "\" characters with "\"
:upper	RFC 5229			Convert lower case characters to upper case
:upperfirst	RFC 5229			Convert first character to upper case if it is a letter and lower case; rest of extracted string left unchanged
fileinto <i>folder-name</i>				
	RFC 3028	max_fileintos MTA option, fileinto and flagtransfer channel options	require "fileinto";	Deliver into specified folder
:copy	RFC 3894		require "copy";	Modifies the <code>fileinto</code> and <code>redirect</code> actions so that the action is performed without affecting Sieve's "implicit keep"
:flags	RFC 5232		require "imap4flags";	Used with <code>keep</code> or <code>fileinto</code> to deliver the message with exactly the specified IMAP flag(s)
:owner	Private			Do the "fileinto" to the Sieve owner's address, rather than to the user's address; cases where the Sieve owner is different than the user on whose behalf the Sieve is being applied (hence cases where this argument would be relevant) might be "head of household" Sieves or system Sieves, when one might want to perform a "fileinto" to a folder belonging to the "head of household" or <code>postmaster</code> , respectively, rather than to a folder belonging to the original message recipient
hold	Private	system-level+		When specified in a system-level Sieve script, sideline a message as a <code>.HELD</code> file in the MTA queue area; ignored (no error) in user-level Sieve scripts
importanceadjust <i>value-string</i>				
	Private			Set or adjust a message's importance
jettison [<i>log-string</i>]				
	Private	filter_jettison and log_filter MTA options, logging channel option	require "jettison";	Non-overrideable discard; a <i>log-string</i> may be specified to be included in the log_filter field in MTA message transaction log entries
keep	RFC 3028			Deliver message "normally"
:flags	RFC 5232		require "imap4flags";	Used with <code>keep</code> or <code>fileinto</code> to deliver the message with exactly the specified IMAP flag(s)
memcache	Private	memcache_host , enable_sieve_memcache MTA options		(New in MS 8.0) Used as an action to manipulate data via the memcache protocol; used as a test to access data via the memcache protocol
:add [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [:duplicate] [:timeout <i>timeout-number</i>] <i>key-string value-string</i>				
	Private			Add a new entry with the specified key, value, and timeout. The operation only succeeds if the entry does not already exist. The operation returns TRUE if the entry is added successfully, FALSE if it is not.
:adjustdown <i>value</i> [:host <i>host-str</i>] [:tableprefix <i>prefix-str</i>] [:duplicate] [:quotatimeout <i>quotatimeout-num</i>] [:penalize] [:timeout <i>timeout-num</i>] <i>key-str</i>				
	Private			Decrement the entry with the specified key by the specified adjustment value.
:adjustup <i>value</i> [:host <i>host-str</i>] [:tableprefix <i>prefix-str</i>] [:duplicate] [:quotatimeout <i>quotatimeout-num</i>] [:penalize] [:timeout <i>timeout-num</i>] <i>key-str</i>				
	Private			Increment the entry with the specified key by the specified adjustment value.
:append [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [:duplicate] <i>key-string value</i>				
	Private			Append the specified value to the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.)
:fetch [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] <i>key-string</i>				

Brief overview of Sieve language elements

	Private			Fetches the value of the entry with the specified key, or return an empty string if the entry doesn't exist.
<code>:prepend[:host <i>host-string</i>][:tableprefix <i>prefix-string</i>][:duplicate] <i>key-string</i> <i>value-string</i></code>				
	Private			Prepend the specified value to the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.)
<code>:remove[:host <i>host-string</i>][:tableprefix <i>prefix-string</i>][:duplicate][:lockout <i>lockout-numeric</i>] <i>key-string</i></code>				
	Private			Remove the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.)
<code>:replace[:host <i>host-string</i>][:tableprefix <i>prefix-string</i>][:timeout <i>timeout-value</i>] <i>key-string</i> <i>value-string</i></code>				
	Private			Update the value and timeout of an (existing) entry. (Returns TRUE if the operation is successful, FALSE if it is not.)
<code>:store[:host <i>host-string</i>][:tableprefix <i>prefix-string</i>][:timeout <i>timeout-value</i>] <i>key-string</i> <i>value-string</i></code>				
	Private			Creates a new entry or updates an existing entry with the specified key, setting the entry to have the specified value and timeout. (Returns TRUE if the operation is successful, FALSE if it is not.)
<code>metermaid</code>	Private	<code>enable_sieve_metermaid</code> MTA option		(New in MS 8.0) Used as an action to manipulate data stored in MeterMaid; used as a test to access data stored in MeterMaid
<code>nonotify</code>	Private	system-level		System-level Sieve action to suppress all applications of either form of <code>notify</code> action
<code>notify[:method "email"][:id <i>string</i>][:options <i>recipient-addr</i> [<i>subject-str</i>] [<:<i>days</i> / <i>hours</i> / <i>seconds</i> > <i>num</i>] [:<i>low</i> / <i>normal</i> / <i>high</i>] [:<i>message</i> <i>message-str</i>]</code>				
	<code>draft-martin-sieve-notify-01</code>	<code>max_notifys</code> MTA option	<code>require "notify";</code>	Generate a new message, typically some form of notification message
<code>:high</code>	<code>draft-...-01</code>			Urgent priority
<code>:id</code>	<code>draft-...-01</code>			
<code>:low</code>	<code>draft-...-01</code>			Non-urgent priority
<code>:method</code>	<code>draft-...-01</code>			The type of notification to generate: only "email" is supported
<code>:normal</code>	<code>draft-...-01</code>			Normal priority
<code>:options</code>	<code>draft-...-01</code>			:method-specific options; for the "email" method, the option value is the recipient email address to which to send the generated notification
<code>:echo</code>	Private			
<code>:mime</code>	Private			
<code>:days</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> , and <code>notify_timeout_default</code> MTA options		Specify duplicate notification timeout in units of days
<code>:hours</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> , and <code>notify_timeout_default</code> MTA options		Specify duplicate notification timeout in units of hours
<code>:seconds</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> , and <code>notify_timeout_default</code> MTA options		Specify duplicate notification timeout in units of seconds
<code>notify[:from <i>string</i>][:importance <"1" / "2" / "3" >][:options <i>string-list</i>] [:<i>message</i> <i>string</i>] <i>method-string</i></code>				
	RFC 5435	<code>max_notifys</code> , <code>notify_ignore_errors</code> MTA options	<code>require "enotify";</code>	Generate a new message, typically some form of notification message
<code>:from</code>	RFC 5435			Specify author (From address) of notification
<code>:importance</code>	RFC 5435			Specify priority, where 1 corresponds to urgent priority, 2 corresponds to normal priority, and 3 corresponds to non-urgent priority; the default is 2 (normal)
<code>:options</code>	RFC 5435			
<code>:message</code>	RFC 5435			
<code>:echo</code>	Private			
<code>:mime</code>	Private			
<code>:days</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> , and <code>notify_timeout_default</code> MTA options		Specify duplicate notification timeout in units of days
<code>:hours</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> ,		Specify duplicate notification timeout in units of hours

Brief overview of Sieve language elements

		and <code>notify_timeout_default</code> MTA options		
<code>:seconds</code>	Private	<code>notify_minimum_timeout</code> , <code>notify_maximum_timeout</code> , and <code>notify_timeout_default</code> MTA options		Specify duplicate notification timeout in units of seconds
<code>novacation</code>	Private	system-level		System-level Sieve action to suppress all applications of the <code>vacation</code> action
<code>override</code>	Private		require "override";	(New in MS 8.0) Mark this Sieve script as determining the disposition of a message
redirect [MODIFIERS] <i>address</i>				
	RFC 3028	<code>max_redirects</code> , <code>max_redirect_addresses</code> MTA options		"Forward" message to specified recipient address(es)
<code>:copy</code>	RFC 3894		require "copy";	Modifies the <code>fileinto</code> and <code>redirect</code> actions so that the action is performed without affecting Sieve's "implicit keep"
<code>:keepmailfrom</code>	Private			Retain original envelope FROM address on <code>redirect</code> ; <code>:keepmailfrom</code> is the default unless <code>:notify</code> is specified in which case the default switches to <code>:resetmailfrom</code>
<code>:list</code>	RFC 6134	SIEVE_EXTLISTS mapping table, <code>max_redirect_addresses</code> MTA option	require "extlists";	Externally stored list of addresses to which to forward with <code>redirect</code> action
<code>:noresent</code>	Private	<code>sieve_redirect_add_resent</code> MTA option		Do not add Resent-?: header lines upon <code>redirect</code>
<code>:notify</code>	RFC 6009		require "redirect-dsn";	Set message's NOTIFY parameter on <code>redirect</code>
<code>:resent</code>	Private	<code>sieve_redirect_add_resent</code> MTA option		Add Resent-?: header lines upon <code>redirect</code>
<code>:resetmailfrom</code>	Private			Reset envelope FROM address to <code>Sieve owner</code> on <code>redirect</code>
<code>:ret</code>	RFC 6009		require "redirect-dsn";	Set message's NOTIFY RET (return-of-content) parameter on <code>redirect</code>
refuse <i>string</i>				
	Private	++	require "refuse";	Refuse message, at SMTP level if possible, falling back to generating MDN
reject <i>string</i>				
	RFC 3028, RFC 5429		require "reject";	Refuse message
removeconversiontag <i>string-or-list</i>				
	Private	system-level		System-level Sieve action to remove the specified <code>conversion tag(s)</code> from the message
removeflag [<i>variable-name</i>] <i>list-of-flags</i>				
	RFC 5232	(<code>max_variables</code> MTA option, if using optional <code>variable</code> argument)	require "imap4flags"; (or require ["imap4flags", "variables"]); if using optional <code>variable</code> argument)	Remove the specified IMAP flag(s) from the message
replaceheader [: <i>index value</i> [: <i>last</i>]] [: <i>newname header-field-name-str</i>] [: <i>newvalue string</i>] [COMPARATOR] [MATCH-TYPE] <i>header-field-name-str</i> [<i>value-patterns-list</i>]				
	<code>draft-degenerate-editheader-00</code>		require "editheader";	Replace a header line
<code>:index</code>	<code>draft-...-00</code>			Separate though equivalent to the RFC 5260 " <code>:index</code> " MATCH-TYPE: attempt to match (and hence potentially replace) only exactly the specified occurrence of the specified header line, by default starting counting from the beginning of the message header
<code>:last</code>	<code>draft-...-00</code>			For the " <code>:index</code> " modifier, count backwards from the end of the message header
<code>:newname</code>	<code>draft-...-00</code>			Change the name of all matching header fields to the specified new name
<code>:newvalue</code>	<code>draft-...-00</code>			Change the value of all matching header fields to the specified new value
set [MODIFIERS] <i>variable-name string-value</i>				
	RFC 5229	<code>max_variables</code> MTA option	require "variables";	Set a variable to a value
<code>:encodeurl</code>	RFC 5435			In a <code>variables</code> set command, perform percent-encoding (as per RFC 3986) of the string value

Brief overview of Sieve language elements

<code>:length</code>	RFC 5229			Decimal number of characters in the value string, converted to a string
<code>:lower</code>	RFC 5229			Convert upper case characters to lower case
<code>:lowerfirst</code>	RFC 5229			Convert first character to lower case if it is a letter and upper case; rest of string left unchanged
<code>:quoteregex</code>	Private			In a <code>variables</code> set command, backslash quote any characters requiring quoting for regex, namely asterisk, question mark, or backslash (*, ?, \)
<code>:quotewild</code>	Private			Prefix "!", "?", "\" characters with "\"
<code>:quotewildcard</code>	RFC 5229			Prefix "!", "?", "\" characters with "\"
<code>:upper</code>	RFC 5229			Convert lower case characters to upper case
<code>:upperfirst</code>	RFC 5229			Convert first character to upper case if it is a letter and lower case; rest of string left unchanged
setconversiontag <i>string-or-list</i>				
	Private	system-level		System-level Sieve action to set the message to have exactly the specified conversion tag(s)
setenvelopefrom <i>address-string-or-list</i>				
	Private	system-level		(New in MS 8.0) System-level Sieve action to override a message's original envelope From address
setflag [<i>variable-name</i>] <i>list-of-flags</i>				
	RFC 5232	(max_variables MTA option if using <code>variables</code> ; with LMTP delivery, see also the flagtransfer channel option)	require "imap4flags"; (or require ["imap4flags", "variables"]); if using <code>variables</code>)	Set the message to have exactly the specified IMAP flag(s)
setmtpriority <i>integer-or-string</i>				
	Private	system-level		(New in MS 8.0) System-level Sieve action to set a message's MT-PRIORITY
setnotify <i>string-or-list</i>				
	Private	system-level		System-level Sieve action to set the DSN NOTIFY parameter (defined in RFC 3461); the value may be the string "NEVER", or one (or a list) of the strings "FAILURE", "SUCCESS", "DELAY"
setoperation < "SUBMIT" / "PASSTHROUGH" / "RELAY" / "DEFAULT" >				
	Private	system-level		(New in MS 8.0) System-level Sieve per-recipient override of type of enqueue operation being performed
setpriority <i>string-or-numeric-priority-value</i>				
	Private	system-level		System-level Sieve action to override message's effective processing priority
setreturn < "FULL" / "HDRS" / "HEADERS" >				
	Private	system-level		System-level Sieve action to set the DSN RET parameter (defined in RFC 3461)
spamadjust <i>numeric-value-string</i>				
	Private			Set a message's "spam level"
transactionlog <i>string-or-list</i>				
	Private	system-level, log_transactionlog MTA option, logging channel option		(New in MS 8.0) System-level Sieve action specifying additional string to include in the MTA message transaction log file
vacation [: <i>days number</i>] [: <i>subject string</i>] [: <i>from address</i>] [: <i>addresses string-list</i>] [: <i>mime</i>] [: <i>handle string</i>] <i>reason-string</i>				
	RFC 5230	non-system-level, max_vacations and vacation_template MTA options	require "vacation";	Generate a vacation auto-response (an "I'm on vacation" sort of message)
<code>:addresses</code>	RFC 5230			Additional addresses to consider as "belonging" to the user, and hence whose presence on a recipient header line of an original message satisfies one requirement for vacation message generation
<code>:days</code>	RFC 5230	autoreply_timeout_default , vacation_minimum_timeout , and vacation_maximum_timeout MTA options		Specify timeout in days
<code>:echo</code>	Private			Modifier on <code>vacation</code> to produce a "processed" MDN response
<code>:from</code>	RFC 5230			Address to use in the From: header line of the generated vacation message

Brief overview of Sieve language elements

<code>:handle</code>	RFC 5230			Explicit label for this vacation operation, to be used instead of the usual argument-based label, so that vacation actions with different arguments may have a coordinated label and thereby respect each other's prior execution (and not generate yet another vacation message)
<code>:headers</code>	Private			Modifier on <code>vacation</code> to produce a response containing header lines, rather than the default MDN called for by the standard
<code>:hours</code>	Private	autoreply_timeout_default , vacation_minimum_timeout , and vacation_maximum_timeout MTA options		Specify the <code>vacation</code> action's autoreponse period in hours, rather than the normal days
<code>:mime</code>	RFC 5230			The <i>reason-string</i> is a MIME entity, including MIME headers as well as content
<code>:noaddresses</code>	Private			Modifier on <code>vacation</code> to suppress the MTA'S normal requirement (as per RFC 5230 , Section 4.5) to only respond if the recipient address or one of its aliases appears explicitly on a recipient header line
<code>:reply</code>	Private			Modifier on <code>vacation</code> to produce a pure reply, containing only the reply text, rather than the default MDN called for by the standard
<code>:seconds</code>	RFC 6131	autoreply_timeout_default , vacation_minimum_timeout , and vacation_maximum_timeout MTA options	require "vacation-seconds";	Specify <code>vacation</code> action's autoreponse period in seconds, rather than the normal days
<code>:subject</code>	RFC 5230			Text to use in Subject: header line of generated vacation message
virusset <i>numeric-value-string</i>				
	Private			Set a message's "virus level"
warn <i>string-or-list</i>				
	Private	system-level, log_filter MTA option, logging channel option		(New in MS 8.0) System-level Sieve action specifying (an additional) string to appear in the "warn" clause in the log_filter field of MTA message transaction log entries
Tests				
address [ADDRESS-PART] [COMPARATOR] [MATCH-TYPE] <i>header-list value-list</i>				
	RFC 3028			Match Internet address in structured headers
<code>:aindex</code>	Private			Match against Nth address on header line
<code>:index</code>	RFC 5260		require "index";	Match against Nth named header line for address , header , or date tests
<code>:last</code>	RFC 5260		require "index";	Used with " :index to match against Nth named header line, counting backwards, for address , header , or date tests
:mime [:anychild] [MIMEOPTS]				
	RFC 5703		require "mime";	Alter address , exists , or header tests to check aspects of structured MIME header lines
<code>:anychild</code>	RFC 5703		require "mime";	Modify <code>:mime</code> to look inside any nested parts
<code>:raw</code>	Private analogue of RFC 5173			(New in MS 7.0.5) Do not MIME decode any RFC 2047 MIME encoded-words
<code>:text</code>	Private analogue of RFC 5173			(New in MS 7.0.5) Perform MIME decoding of any RFC 2047 MIME encoded-words
<code>allof test-list</code>	RFC 3028			Logical AND of tests; <i>test-list</i> has the form (<i>test</i> [, <i>test</i>]*)
<code>anyof test-list</code>	RFC 3028			Logical OR of tests
body [COMPARATOR] [MATCH-TYPE] [BODY-TRANSFORM] <i>value-list</i>				
	RFC 5173	enable_sieve_body MTA option	require "body";	Match content in the body of an email message; (only supports <code>:contains</code> and <code>:isMATCH-TYPES</code> , and a limited set of COMPARATORS)
currentdate [:zone <i>time-zone</i>] [COMPARATOR] [MATCH-TYPE] <i>date-part value-list</i>				
	RFC 5260		require "date";	Compare against current date-time
<code>:zone</code>	RFC 5260		require "date";	Specify a time zone offset to which to shift the current date-time prior to testing
date [:zone <i>time-zone</i> / :originalzone] [COMPARATOR] [MATCH-TYPE] <i>header-name date-part value-list</i>				
	RFC 5260		require "date";	Match against date from a header line
<code>:originalzone</code>	RFC 5260		require "date";	Test retaining the time zone offset of the original date time
<code>:zone</code>	RFC 5260		require "date";	Specify a time zone offset to which to shift the date-time prior to testing
duplicate [:handle <i>string</i>] [:header <i>header-name</i> / :uniqueid <i>value</i>] [:seconds <i>timeout</i>] [:last]				

Brief overview of Sieve language elements

	draft-bosch-sieve-duplicate-09	max_duplicates , duplicate_tracking_url MTA options	require "duplicate";	(New in MS 8.0) Test whether "the same" message was (recently) already received
<code>:handle</code>	draft-bosch-sieve-duplicate-09		require "duplicate";	Distinguish, via handle name, this duplicate test from other duplicate tests
<code>:header</code>	draft-bosch-sieve-duplicate-09		require "duplicate";	Use as unique ID (for duplicate detection) the content of the specified header field
<code>:last</code>	draft-bosch-sieve-duplicate-09	duplicate_timeout_default	require "duplicate";	Modify <code>:seconds</code> interpretation (or if <code>:seconds</code> was not specified, use duplicate_timeout_default value) interpreting as seconds within which a "duplicate" message must have been received (counting from the most recent check of the same unique ID) to count as a duplicate
<code>:mime</code>	Private		require "mime";	
<code>:anychild</code>	Private		require "mime";	
<code>:seconds</code>	draft-bosch-sieve-duplicate-09	duplicate_timeout_default , duplicate_minimum_timeout , duplicate_maximum_timeout MTA options	require "duplicate";	Seconds within which a "duplicate" message must have been received (counting time from the first message with the same unique ID) to count as duplicate
<code>:uniqueid</code>	draft-bosch-sieve-duplicate-09		require "duplicate";	Use the specified value as the unique ID (for duplicate detection)
envelope [COMPARATOR] [ADDRESS-PART] [MATCH-TYPE] <i>envelope-part-list value-list</i>				
	RFC 3028		require "envelope";	Match SMTP envelope address
environment [COMPARATOR] [MATCH-TYPE] <i>environment-item value-list</i>				
	RFC 5183		require "environment";	Match against operating environment information
exists <i>header-list value-list</i>				
	RFC 3028			Test whether specified header(s) are present in a message
<code>:list</code>	Private enhancement of RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in <code>address</code> , <code>envelope</code> , and <code>header</code> tests; MTA private additional functionality is that <code>:list</code> is also supported on the standard <code>exists</code> test, as well as on extension tests <code>currentdate</code> , <code>date</code> , <code>environment</code> , <code>hasflag</code> , <code>spamtest</code> , <code>string</code> , and <code>virustest</code> , and on <code>deleteheader</code> and <code>replaceheader</code> actions
<code>:mime</code> [<code>:anychild</code>] [MIMEOPTS]				
	RFC 5703		require "mime";	Alter <code>address</code> , <code>exists</code> , or <code>header</code> tests to check aspects of structured MIME header lines
<code>:anychild</code>	RFC 5703		require "mime";	Modify <code>:mime</code> to look inside any nested parts
<code>:regex</code>	Private enhancement of draft-murchison-sieve-regex-08	enable_sieve_regex MTA option	require "regex";	Regex match type
<code>false</code>	RFC 3028			Always evaluate to FALSE
hasflag [MATCH-TYPE] [COMPARATOR] [<i>variable-list</i>] <i>list-of-flags</i>				
	RFC 5232	(max_variables MTA option if using optional <code>variable</code> argument)	require "imap4flags"; (or require ["imap4flags", "variables"]); if using optional <code>variable</code> argument)	Test whether the message has the specified IMAP flag(s)
<code>:list</code>	Private enhancement of RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in <code>address</code> , <code>envelope</code> , and <code>header</code> tests; MTA private additional functionality is that <code>:list</code> is also supported on the standard <code>exists</code> test, as well as on extension tests <code>currentdate</code> , <code>date</code> , <code>environment</code> , <code>hasflag</code> , <code>spamtest</code> , <code>string</code> , and <code>virustest</code> , and on <code>deleteheader</code> and <code>replaceheader</code> actions
header [COMPARATOR] [MATCH-TYPE] <i>header-list value-list</i>				
	RFC 3028	defer_header_addition and (new in MS 8.0) sieve_received MTA options		Match header
<code>:mime</code> [<code>:anychild</code>] [MIMEOPTS]				
	RFC 5703		require "mime";	Alter <code>address</code> , <code>exists</code> , or <code>header</code> tests to check aspects of structured MIME header lines

<code>:anysub</code>	RFC 5703		require "mime";	Modify <code>:mime</code> to look inside any nested parts
<code>:raw</code>	Private analogue of RFC 5173			(New in MS 7.0.5) Do not MIME decode any RFC 2047 MIME encoded-words
<code>:text</code>	Private analogue of RFC 5173			(New in MS 7.0.5) Perform MIME decoding of any RFC 2047 MIME encoded-words
<i>ihave capability-list</i>				
	RFC 5463		require "ihave";	Test which Sieve extensions are available
<code>importancetest</code> [COMPARATOR] [MATCH-TYPE] <i>value</i>				
	Private			Test a message's importance
<code>memcache</code>	Private	<code>memcache_host</code> , <code>enable_sieve_memcache</code> MTA options		(New in MS 8.0) Used as an action to manipulate data via the memcache protocol; used as a test to access data via the memcache protocol
<code>memcache :adjustdown</code> <i>value</i> [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [:duplicate] [MATCH-TYPE] [COMPARATOR] [:quotatimeout <i>quotatimeout-numeric</i> [:penalize]] [:timeout <i>timeout-value</i>] <i>key-string</i> [test-string]				
	Private			For the entry with the specified key, adjust the value down as specified and compare that adjusted value against the specified <i>test-string</i> value, which same <i>test-string</i> then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to 'value "lt"' and 'comparator "i;ascii-numeric"', respectively.
<code>memcache :adjustup</code> <i>value</i> [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [:duplicate] [MATCH-TYPE] [COMPARATOR] [:quotatimeout <i>quotatimeout-numeric</i> [:penalize]] [:timeout <i>timeout-value</i>] <i>key-string</i> [test-string]				
	Private			For the entry with the specified key, adjust the value up as specified and compare that adjusted value against the specified <i>test-string</i> value, which same <i>test-string</i> then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to 'value "gt"' and 'comparator "i;ascii-numeric"', respectively.
<code>memcache :fetch</code> [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [MATCH-TYPE] [COMPARATOR] <i>key-string</i> [test-string]				
	Private			For the entry with the specified key, compare its current value against the <i>test-string</i> , setting the value to <i>test-string</i> . The test always fails if the entry does not exist. MATCH-TYPE and COMPARATOR default to 'is' and 'i;ascii-casemap', respectively.
<code>memcache :throttle</code> :quota <i>quota-numeric</i> [:duplicate] :quotatimeout <i>quotatimeout-numeric</i> [:penalize] [:host <i>host-string</i>] [:tableprefix <i>prefix-string</i>] [:timeout <i>timeout-value</i>] [MATCH-TYPE] [COMPARATOR] [:adjustup <i>adjustment-value</i>] [:adjustdown <i>adjustment-value</i>] <i>key-string</i> [test-string]				
	Private			Implement MeterMaid throttle capability. The throttle value is incremented by default, though "adjustup" or "adjustdown" may be used to specify a non-default adjustment of the value. Then if none of MATCH-TYPE, COMPARATOR, nor <i>test-string</i> are specified, the test returns TRUE if the throttle is engaged, or FALSE if it is not engaged (or an error occurs). If any of the MATCH-TYPE, COMPARATOR, or <i>test-string</i> parameters is specified, then the throttle entry's value is adjusted and then the specified Sieve test is applied to the value. The default MATCH-TYPE is 'value "gt"' and the default COMPARATOR is 'i;ascii-numeric'.
<code>metermaid</code>	Private	<code>enable_sieve_metermaid</code> MTA option		(New in MS 8.0) Used as an action to manipulate data stored in MeterMaid; used as a test to access data stored in MeterMaid
<code>:adjustdown</code> <i>adjustment-value</i> [:host <i>host-string</i>] [:duplicate] [MATCH-TYPE] [COMPARATOR] <i>table-string</i> <i>key-string</i> [test-string]				
	Private			Decrement the entry with the specified key in the specified table by the specified adjustment value.
<code>:adjustup</code> <i>adjustment-value</i> [:host <i>host-string</i>] [:duplicate] [MATCH-TYPE] [COMPARATOR] <i>table-string</i> <i>key-string</i> [test-string]				
	Private			Increment the entry with the specified key in the specified table by the specified adjustment value.
<code>:fetch</code> [:host <i>host-string</i>] [MATCH-TYPE] [COMPARATOR] <i>table-string</i> <i>key-string</i> [test-string]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor <i>test-string</i> is specified, the specified entry's value is returned as a string, or an empty string is returned if the specified entry doesn't exist. If any of MATCH-TYPE, COMPARATOR, or <i>test-string</i> is specified, test against the current value of the specified entry (MATCH-TYPE and COMPARATOR default to 'is' and 'comparator "i;ascii-casemap"', respectively), returning FALSE if the specified entry does not exist.
<code>:greylisting</code> [:host <i>host-string</i>] [:duplicate] [MATCH-TYPE] [COMPARATOR] <i>table-string</i> <i>key-string</i> [test-string]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor <i>test-string</i> are specified, then return TRUE if greylisting is engaged, while FALSE is returned if greylisting is not engaged or an error occurs. If any of MATCH-TYPE, COMPARATOR, or <i>test-string</i> is specified (MATCH-TYPE and COMPARATOR default to 'is' and 'comparator "i;ascii-

Brief overview of Sieve language elements

				casemap", respectively), then increment the throttle and then apply the Sieve test to the value.
<code>:remove</code> [:host <i>host-string</i>][:duplicate] <i>table-string key-string</i>				
	Private			Remove the entry with the specified key from the specified table. Returns TRUE if the operation is successful; FALSE if it is not.
<code>:store</code> [:host <i>host-string</i>] <i>table-string key-string value-string</i>				
	Private			Creates a new entry or updates an existing entry with the specified key in the specified table. Returns TRUE if the operation is successful; FALSE if it is not.
<code>:throttle</code> [:host: <i>host-string</i>][:duplicate] [MATCH-TYPE] [COMPARATOR] <i>table-string key-string</i>				
	Private			MeterMaid throttle functionality. If none of MATCH-TYPE, COMPARATOR, nor <i>test-string</i> is specified, return TRUE if the throttle is engaged, or FALSE if the throttle is not engaged or an error occurs; if any of these three parameters is specified, then increment the throttle and then apply the Sieve test to the value. (MATCH-TYPE defaults to ':value "gt"' and COMPARATOR defaults to ':comparator "i;ascii-numeric"')
<code>not test</code>	RFC 3028			Reverse value of test argument
<code>notify_method_capability</code> [COMPARATOR] [MATCH-TYPE] <i>notification-uri notification-capability value-list</i>				
	RFC 5435		require "enotify";	Test whether a capability of a method of notification is available to be performed as desired
<code>:list</code>	Private enhancement of RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in address, envelope, and header tests; MTA private additional functionality is that ":list" is also supported on the standard exists test, as well as on extension tests <code>currentdate</code> , <code>date</code> , <code>environment</code> , <code>hasflag</code> , <code>notify_method_capability</code> , <code>spamtest</code> , <code>string</code> , and <code>virustest</code> , and on <code>deleteheader</code> and <code>replaceheader</code> actions
<code>size</code> < :over / :under > <i>limit-value</i>				
	RFC 3028			Test size of message
<code>:over</code>	RFC 3028			Match if message size is over specified value
<code>:under</code>	RFC 3028			Match if message size is under specified value
<code>spamtest</code> [COMPARATOR] [MATCH-TYPE] <i>value</i>				
	RFC 3685		require "spamtest"; or require "spamtestplus";	Test a message's "spam level"
<code>:list</code>	RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in address, envelope, and header tests; MTA private additional functionality is that ":list" is also supported on the standard exists test, as well as on extension tests <code>currentdate</code> , <code>date</code> , <code>environment</code> , <code>hasflag</code> , <code>notify_method_capability</code> , <code>spamtest</code> , <code>string</code> , and <code>virustest</code> , and on <code>deleteheader</code> and <code>replaceheader</code> actions
<code>:percent</code>	RFC 5235		require "spamtestplus";	Test a message's "spam level" as a percentage value
<code>string</code> [MATCH-TYPE] [COMPARATOR] <i>source-string-list value-list</i>				
	RFC 5229	<code>max_sieve_string_size</code> MTA option	require "variables";	Test the (string) value of <code>variable(s)</code>
<code>:list</code>	Private enhancement of RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in address, envelope, and header tests; MTA private additional functionality is that ":list" is also supported on the standard exists test, as well as on extension tests <code>currentdate</code> , <code>date</code> , <code>environment</code> , <code>hasflag</code> , <code>spamtest</code> , <code>string</code> , and <code>virustest</code> , and on <code>deleteheader</code> and <code>replaceheader</code> actions
<code>true</code>	RFC 3028			Always evaluate to TRUE
<code>valid_ext_list</code> <i>extlist-names-list</i>				
	RFC 6134		require "extlists";	Test whether an <code>external list</code> specification is valid (both syntactically and semantically)
<code>valid_notify_method</code> <i>notification-uris-list</i>				
	RFC 5435		require "enotify";	Test whether <code>notification methods</code> are supported and syntactically valid
<code>virustest</code> [COMPARATOR] [MATCH-TYPE] <i>value</i>				
	RFC 3685		require "virustest";	Test a message's "virus level"

<code>:count</code>	RFC 5231		require "relational";	Match type for numeric comparison on number of occurrences
<code>:list</code>	RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in <code>address</code> , <code>currentdate</code> , <code>date</code> , <code>envelope</code> , <code>exists</code> , and <code>header</code> tests, as well as in the <code>string</code> test when the <code>variables</code> extension is enabled, and in <code>spamtest</code> and <code>virustest</code> tests when those extensions are enabled
<code>:regex</code>	draft-murchison-sieve-regex-08	enable_sieve_regex MTA option	require "regex";	Regex match type
<code>:value</code>	RFC 5231		require "relational";	Match type for comparison on string values
<code>test-name string</code>	Private	FILTER_test-name mapping table		Custom test <code>test-name</code> : probes mapping table <code>FILTER_test-name</code> with <code>string</code> , evaluating to TRUE if mapping table sets <code>\$Y</code> flag, or FALSE otherwise
Functions				
<code>translate string-or-list source-charset dest-charset</code>				
	Private			(New in MS 8.0.1) Perform character set conversion , returning translated string or list
<code>test-name string</code>	Private	FILTER_test-name mapping table	require "variables";	Custom test <code>test-name</code> used as a function: probes mapping table <code>FILTER_test-name</code> with <code>string</code> , returning mapping's string result in variable <code>#{0}</code> and mapping's flag result in variable <code>#{1}</code>
<code>...many-other-functions...</code>	Private			See Symbol table functions
Values				
<code>non-negative-integer</code>	RFC 3028			Non-negative integer, range 0 to 2,147,483,647 (2 ³¹ - 1); suffixes "K" (1024 or 2 ¹⁰), "M" (2 ²⁰), or "G" (2 ³⁰) are permitted
<code>negative-integer</code>	Private			Negative integers are also supported by the MTA
<code>"utf-8-string"</code>	RFC 3028	max_sieve_string_size MTA option		String of characters represented in UTF-8; note that non-printing characters such as CR and LF are permitted, and so in particular strings can span multiple lines
<code>"\character"</code>	RFC 3028			<code>\\</code> represents the backslash character, <code>\</code> represents the double quote character, and otherwise the backslash is ignored as if it were not present
<code>text:multiline text ended by CRLF.CRLF</code>	RFC 3028			More convenient way to enter multiline text strings; note must dot-stuff (as with SMTP messages)
<code>["s1", "s2", ..., "sN"]</code>	RFC 3028	max_sieve_list_size MTA option		List of strings
<code>"\${hex:hex-pair-sequence}"</code>	RFC 5228		require "encoded-character";	<code>hex-pair-sequence</code> is one or more hexadecimal-encoded characters, space separated; e.g., <code>"{hex: 40 24}"</code> represents <code>"@\${"</code>
<code>"\${unicode: unicode-hex-sequence}"</code>	RFC 5228		require "encoded-character";	<code>unicode-hex-sequence</code> is one or more Unicode hex-encoded characters, space separated; e.g., <code>"\${unicode: 000040}"</code> represents <code>"@"</code>
<code>\$(variable-name)</code>	RFC 5229	max_variables MTA option	require "variables";	<code>variable-name</code> may be a variable identifier, or digit(s) identifying the variable, e.g., <code>#{a}</code> refers to the variable named "a", while <code>#{0}</code> refers to the 0th variable; variable namespaces are not currently supported (so in particular variable names may not include any periods); note that variable substitutions are <i>not</i> supported in "body" test arguments
<code>expression</code>	Private			The MTA supports a variety of string and numeric operators and functions, and supports expressions to compute values; see Arithmetic operators and Symbol table functions and Symbol table operators
ADDRESS-PARTS				
<code>:all</code>	RFC 3028			Entire address
<code>:comment</code>	Private			Extract any parenthetical comment appearing after an address
<code>:detail</code>	RFC 3598		require "subaddress";	Extract subaddress (e.g., folder name) in <code>address</code> and <code>envelope</code> tests
<code>:display</code>	Private			Extract the display-name phrase (the optional text appearing outside of and specifically in front of the actual address, the actual address being the part enclosed in angles)
<code>:domain</code>	RFC 3028			Domain portion of address: the portion to the right of the @ character
<code>:localpart</code>	RFC 3028			RFC 822 "local-part" of address: the portion to the left of the @ character

Brief overview of Sieve language elements

<code>:user</code>	RFC 3598		require "subaddress";	Extract username (<i>i.e.</i> , local-part minus subaddress) in address and envelope tests
BODY-TRANSFORMS				
<code>:raw</code>	RFC 5173		require "body";	Interpret message body as an unprocessed "blob"; in particular, do not interpret MIME structure nor decode any content-transfer-encoding
<code>:text</code>	RFC 5173		require "body";	Decode MIME encoded text
COMPARATORS				
<code>:comparator "i:ascii-casemap"</code>	RFC 3028 , RFC 4790			Treat uppercase and lowercase characters in the ASCII subset of UTF8 the same
<code>:comparator "i:ascii-casemap-collapse"</code>	Private		require "comparator-i:ascii-casemap-collapse";	(New in MS 8.0) Similar to "i:ascii-casemap", except that all folding white space characters (space, tab, carriage return, line feed) are removed from both the target and pattern strings prior to comparison; this comparator is recommended for working around standards-incompliant client behavior regarding white space, folding white space, and MIME encoded-words
<code>:comparator "i:ascii-integer"</code>	Private		require "comparator-i:ascii-integer";	(New in MS 8.0) Test signed or unsigned integer values; (not supported on "body" tests)
<code>:comparator "i:ascii-numeric"</code>	RFC 4790		require "comparator-i:ascii-numeric";	Compare arbitrarily sized, unsigned integer numbers stored as octet strings, (in particular, disregard leading zeros, and truncate at the first non-digit character); see RFC 4790, Section 9.1.1 for further details; (not supported on "body" tests)
<code>:comparator "i:octet"</code>	RFC 3028 , RFC 4790			Compare octets
<code>:comparator "i:octet-collapse"</code>	Private		require "comparator-i:octet-collapse";	(New in MS 8.0) Similar to "i:octet", except that all folding white space characters (space, tab, carriage return, line feed) are removed from both the target and pattern strings prior to comparison; this comparator is recommended for working around standards-incompliant client behavior regarding white space, folding white space, and MIME encoded-words
<code>:comparator "i:utf-8"</code>	?		require "comparator-i:utf8";	(New in MS 8.0) (not supported on "body" tests)
DATE-PARTS				
<code>date</code>	RFC 5260		require "date";	Date in "yyyy-mm-dd" format
<code>day</code>	RFC 5260		require "date";	Two digit day, "01", ..., "31"
<code>hour</code>	RFC 5260		require "date";	Two digit hour, "00", ..., "23"
<code>iso8601</code>	RFC 5260		require "date";	Date and time in restricted ISO 8601 format
<code>julian</code>	RFC 5260		require "date";	Modified Julian Day, that is, the date expressed as an integer number of days since 00:00 UTC on November 17, 1958 (using the Gregorian calendar); this corresponds to the Julian Day minus 2400000.5
<code>minute</code>	RFC 5260		require "date";	Two digit minute, "00", ..., "59"
<code>month</code>	RFC 5260		require "date";	Two digit month, "00", ..., "12"
<code>second</code>	RFC 5260		require "date";	Two digit second, "00", ..., "60"
<code>std11</code>	RFC 5260		require "date";	Date and time in a format appropriate for use in a Date: header field (RFC 822 format)
<code>time</code>	RFC 5260		require "date";	Time in "hh:mm:ss" format
<code>weekday</code>	RFC 5260		require "date";	Day of the week expressed in range "0" (Sunday) to "6" (Saturday)
<code>year</code>	RFC 5260		require "date";	Four digit year, "0000", ..., "9999"
<code>zone</code>	RFC 5260		require "date";	Time zone in use, in +hh:mm or -hh:mm format
ENVELOPE-PARTS				
<code>from</code>	RFC 3028		require "envelope";	Envelope From address
<code>to</code>	RFC 3028		require "envelope";	Envelope To address
<code>auth</code>	Private		require ["envelope", "envelope-auth"];	Access message's AUTH value
<code>conversiontag</code>	Private	system-level	require "envelope";	In system-level Sieve, access message's conversion tag(s)

Brief overview of Sieve language elements

envid	RFC 6009		require ["envelope", "envelope-dsn"];	Access message's envelope-id value
orcpt	RFC 6009		require ["envelope", "envelope-dsn"];	Access message's ORCPT value
notify	RFC 6009		require ["envelope", "envelope-dsn"];	Access message's DSN NOTIFY value
ret	RFC 6009		require ["envelope", "envelope-dsn"];	Access message's DSN RET value
ENVIRONMENT-ITEMS				
domain	RFC 5183		require "environment";	Preferably the value of <code>ldap_default_domain</code> MTA option, next the value of <code>received_domain</code> MTA option, or if neither are defined the value of the L channel <code>official_host_name</code>
host	RFC 5183		require "environment";	L channel official-host-name
location	RFC 5183		require "environment";	Type of service evaluating the Sieve; "MTA" for MTA evaluation
name	RFC 5183		require "environment";	"IMTA" for MTA evaluation
phase	RFC 5183		require "environment";	Phase of processing; possible values are "initialize", "connect", "mail" (reported at both HELO and MAIL FROM command stages), "rcptin", "rcptout", "datastart", "dataend", and "pre" (the default value)
remote-host	RFC 5183		require "environment";	Current source system
remote-ip	RFC 5183		require "environment";	Source IP
version	RFC 5183		require "environment";	MTA version string (akin to version reported in Received: header line)
vnd.oracle.last-verdict	Private		require "environment";	(New in MS 7.0.5) Prior Sieve's explicit handling action (see discussion of MTA's Sieve hierarchy)
vnd.oracle.message-hash	Private		require "environment";	(New in MS 8.0) Return any message hash calculated by the <code>generate_message_hash</code> channel option; the query will fail if no message hash was generated
vnd.oracle.mt-priority	Private		require "environment";	(New in MS 8.0) Match against message's current MT-PRIORITY value
vnd.oracle.operation-type	Private		require "environment";	(New in MS 8.0) Match against message's current type of enqueue operation
vnd.oracle.reevaluate	Private		require "environment";	(New in MS 8.0) TRUE or FALSE according to whether this Sieve script is being reevaluated
vnd.oracle.tracking-id	Private		require "environment";	(New in MS 8.0) Return the tracking identifier for the current message; the query will fail if there's no tracking id
vnd.sun.authenticated-sender-address	Private		require "environment";	Match against sender address associated with the authentication for the SMTP session
vnd.sun.authenticated-sender-id	Private		require "environment";	Match against user identity associated with the authentication for the SMTP session
vnd.sun.autoreply-internal	Private		require "environment";	TRUE or FALSE according to whether the autoreply criteria have been met for use of the "internal" autoreply response text stored in LDAP
vnd.sun.destination-channel	Private		require "environment";	Match against message's destination channel
vnd.sun.source-channel	Private		require "environment";	Match against message's source channel
<i>custom-item-name</i>	Private	<code>\$(+Item-name) item-value</code> in an <code>*_ACCESS</code> address mapping table	require "environment";	Arbitrary, custom environment items for a message may be set from the <code>FROM_ACCESS</code> or <code>*_ACCESS</code> recipient access mapping tables, for subsequent use in environment tests
MATCH-TYPES				
<code>:aindex</code>	Private			Match against Nth address on header line
<code>:contains</code>	RFC 3028			Substring match
<code>:count</code>	RFC 5231		require "relational";	Match type for numeric comparison on number of occurrences; (not available for use with "body" test)
<code>:index</code>	RFC 5260		require "index";	Match against Nth named header line for address, header, or date tests
<code>:last</code>	RFC 5260		require "index";	For the <code>:index</code> modifier, count backwards from the end of the message header

Sieve supported extensions

:is	RFC 3028			Exact match
:list	RFC 6134 and private enhancements	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in address, envelope, and header tests; MTA private additional functionality is that ":list" is also supported on the standard exists test, as well as on extension tests currentdate, date, environment, hasflag, importantest, memcache, metermaid, spamtest, string, and virustest, and on deleteheader and replaceheader actions
:matches	RFC 3028			Wildcard match; (not supported on "body" tests)
:regex	draft-murchison-sieve-regex-08	enable_sieve_regex MTA option	require "regex";	Regex match type; (not supported on "hasflag" or "body" tests)
:value	RFC 5231		require "relational";	Match type for comparison on string values; (may not be specified on "body" tests)
MIMEOPTS				
:type	RFC 5703		require "mime";	For Content-type: MIME header field, parse and test the value of the MIME type; for Content-disposition: MIME header field, parse and test the disposition value; for other MIME headers, use a blank string
:subtype	RFC 5703		require "mime";	For Content-type: MIME header field, parse and test the value of the MIME subtype; for other MIME headers, use a blank string
:contenttype	RFC 5703		require "mime";	For Content-type: MIME header field, parse and test the combined value of the MIME type and subtype; for Content-disposition: MIME header field, parse and test the disposition value; for other MIME headers, use a blank string
:param	RFC 5703		require "mime";	Parse the header for MIME parameters, returning true if any value found matches any of the test string values

+Only displayed in `imsimta test -expression -mm` if executing process has "world" privilege

++Prior to MS 6.2, only permitted in system-level Sieves; nowadays supported in arbitrary Sieves

5.1.2 Sieve supported extensions

In addition to the core Sieve functionality specified in [RFC 5228 \(Sieve\)](#), the MTA's Sieve implementation supports a great many standardized extensions, plus many additional private-to-the-MTA extensions. Standardized extensions supported include:

- [Body extension \(RFC 5173\)](#). The body test provides the means to test material in the message body. Note that there are a number of restrictions on the implementation of body. (7U2)
- [Copy extension \(RFC 3894\)](#). Copy is a simple extension that allows the "redirect" and "fileinto" actions to be used without canceling the default action of saving the message to the "inbox". (6.1)
- [Date extension \(RFC 5260\)](#). The date test provides the means to test fields in date-time values. (7U4)
- [Editheader extension \(RFC 5293\)](#). The "addheader" and "deleteheader" actions provide the ability to alter the message header. Additionally, the "replaceheader" action described in [draft-degener-sieve-editheader-00](#) has been implemented. This provides an especially convenient way to add tags to subject fields.
- [Encoded-character extension \(RFC 5228\)](#). This extension provides a way to specify Unicode characters by numeric value in Sieve character strings. Additionally, \r, \n, and \t can be used to represent carriage return, line feed, and tab characters respectively in quoted strings. (7.0)

- [Envelope extension \(RFC 5228\)](#). This extension consists of an "envelope" test that can access MAIL FROM and RCPT TO address information. (Other extensions have made additional items available to the "envelope" test.)
- [Envelope-dsn \(RFC 6009\)](#). This extension provides access to additional envelope information provided by the delivery status notification SMTP extension. (7U2)
- [Environment extension \(RFC 5183\)](#). Environment provides scripts access to information outside of the current message. (7.0U1)
- [Ereject extension \(RFC 5429\)](#). This extension updates the definition of the "reject" action to allow messages to be refused during the SMTP transaction (rather than being accepted and then generating an MDN), and defines the "ereject" action to require messages to be refused during the SMTP transaction. (6.1)
- [Extlists extension \(RFC 6134\)](#). This extension adds the ":list" match-type to the "address", "currentdate", "date", "deleteheader", "envelope", "environment", "hasflag", "header", "replaceheader", "spamtest", "string", and "virustest" tests. ":list" in turn allows the test to check values against externally stored information. (7U1)
- [Fileinto extension \(RFC 5228\)](#). This extension adds the "fileinto" action for specifying a folder where the message is to be delivered.
- [Ihave extension \(RFC 5463\)](#). "ihave" makes it possible to write scripts that use a given extension if it is available but continue to operate if it is not. (7.0)
- [Index extension \(RFC 5260\)](#). The index extension adds a ":index" nonpositional parameter to the address, date, and header tests, which in turn provides the means to check a specific instance of header fields that occur multiple times. (7U4)
- [Imap4flags extension \(RFC 5232\)](#). Imap4flags provides the means to set IMAP flags on messages delivered to the message store. (6.3P1)
- [Mime extension \(from RFC 5703\)](#). The "mime" extension provides facilities for testing headers in inner MIME parts of messages. (7U1) New in 8.0 is support for the "extracttext" and "foreverypart" Sieve extensions from [RFC 5703](#). (8.0)
- [Notify extension \(RFC 5435\) and the mailto notification method \(RFC 5436\)](#). Notify with the mailto method provides the means to send a notification email about the current message being processed. Note that an earlier draft of the "notify" action is also still supported. (6.2, 7.0.5)
- [Relational extension \(RFC 5231\)](#). Relational adds relational comparisons (less than, greater than, *etc.*) to the "header", "address", and "envelope" tests (and other extension tests as they have subsequently been defined). It also adds the ability to count (":count") the number of entities that match the test criteria. (6.0)
- [Redirect-dsn extension \(RFC 6009\)](#). This extends Sieve's "redirect" action to provide control over delivery status notification parameters with two new arguments ":notify" (":notify" support was actually added for 6.3p1, prior to its standardization) and ":ret". (6.3p1, 7U2)
- ["spamtest", "spamtestplus", and "virustest" extensions \(RFC 5235\)](#). The tests defined by these extensions provide a means for Sieve scripts to access spam and virus filter "scores". ("spamtest" and "virustest" 6.0; "spamtestplus" 6.3)

- [Subaddress extension \(RFC 3598\)](#). Subaddress `:user` and `:detail` tagged arguments for the `envelope` and `address` tests to access information [embedded in the local part of an address](#). (iMS uses a [plus sign as the separator between user and detail information in the local part](#).) (6.0)
- [Vacation extension \(RFC 5230\)](#). The `vacation` action defined by this extension can be used in user-level Sieve scripts to generate "out of office" messages in response to incoming email.
- [Vacation-seconds extension \(RFC 6131\)](#). This extends the vacation time to allow specification of timeout values in seconds rather than minutes; in particular, it adds a `:seconds` parameter to the `vacation` action.
- [Variables extension \(RFC 5229\)](#). The core Sieve language does not provide any means of saving state from one statement to the next. This extension adds variables to the language. (6.2)

In addition to the above standardized extensions, the MTA also supports some private extensions:

- The `address` test supports a private `:aindex` tag. This `:aindex` tag accepts a positive integer as an argument. If `:aindex n` is specified, then only the *n*th address in each header field will be tested. Note that `:aindex` (selecting which address out of multiple addresses) may be combined with `:index` (selecting which header line out of multiple header lines), to cause the test to operate on a single address in a single header line. (7.0.5)
- The `address` test supports two new, private, part modifiers, `:display` and `:comment`. The `:display` modifier causes the `address` test to operate on the display-name phrase; note that the display-name phrase is the optional text appearing outside of and specifically in front of the actual address, the actual address being the part enclosed in angles. The `:comment` modifier causes the `address` test to operate on any parenthetical comments that appear after an address. (7.0.5)
- The `addconversiontag`, `removeconversiontag`, and `setconversiontag` actions allow operating on the MTA's private [conversion tag](#) mechanism. (`addconversiontag` and `setconversiontag` 6.0; `removeconversiontag` 7.0u3) Also, in system-level Sieves, the `envelope` test [accepts "conversiontag" as a field specifier value](#), checking the current list of conversion tags, one at a time. (This test only "sees" the set of conversion tags that were present prior to Sieve processing; the effects of `setconversiontag` and `addconversiontag` are not visible.) (6.3)
- The `addprefix` and `addsuffix` actions are available for adding a string argument as text at the beginning or end, respectively, of the first plain text part of a message. (7.0u3)
- The `addtag` action provides a convenient way to add a prefix string, that is, a "tag", to the beginning of the Subject: header line. (6.0)
- The `adjustcounter` action is available for system Sieve filters to manipulate any of the eight signed, 64 bit counters accessible from Sieve filters. (8.0)
- The `capture` action (and the deprecated synonym `monitor`). Two optional nonpositional parameters, `:dsn` and `:message`, were added for JES MS 6.3; `:journal` (to generate Microsoft[®] Exchange "envelope journaling" format) was added for 7.0u2; `:header` (which can be used as a modifier with either `:dsn` or `:journal`) was added for 8.0. (6.3, 7.0u2, 8.0)

- In addition to standard Sieve comparators, the MTA also supports some non-standard [Sieve comparators](#).
- The "debug" action takes a string argument, and makes that string available for logging to a debug log file if MTA debugging is enabled at `mm_debug=2` level or higher.
- The MTA supports the proposed [Sieve duplicate extension](#) specified in [RFC 7352](#). (8.0)
- The [envelope-auth extension](#) adds a "auth" part to the [envelope](#) test, providing access to the SMTP AUTH value.
- The "hold" action causes a message to be sidelined in the MTA queue area as a `.HELD` file.
- The "importancetest" test and "importanceadjust" action allow multiple Sieve scripts to cooperate in making a determination of a message's importance, much like the way that the "spamttest" and "spamadjust" extensions allow multiple Sieve scripts to cooperate in determining whether or not a message is spam.
- The "jettison" action causes a message to be unconditionally discarded; this is a "non-overridable" discard (for use by system-level Sieve scripts).
- The "loop" construct is supported in system-level Sieve scripts. (7.0.5)
- The "memcache" operator permits querying and updating a memcache server from Sieve scripts. (8.0)
- The "metermaid" operator permits querying and updating [MeterMaid](#) from Sieve scripts. (8.0)
- The "override" action is supported in system-level Sieve scripts; the capability string is "override". (8.0)
- The "nonotify" action is supported in system-level Sieve scripts. It suppresses all uses of the "notify" and "enotify" extensions (all applications of either form of the "notify" action). (8.0)
- The "novacation" action is supported in system-level Sieve scripts. (6.1)
- By default, the MTA supports a relaxed use of "require" clauses in Sieve scripts, in that the MTA permits "require" clauses to be sprinkled throughout a Sieve script (rather than, as the Sieve standard specifies, having all "require" clauses at the beginning of the Sieve script). This is especially useful in conjunction with the MTA's support for combining multiple Sieve scriptlets such as those resulting from spam/virus filter package integration, or from user LDAP attribute `mailVacation*` settings. However, enforcement of per-the-Sieve-standard "require" clause location may be selected by setting the [strict_require](#) MTA option.
- The " :raw" and " :text" modifiers defined for the "body" test may now also be used in "header" and "address" tests. Similarly to when used with "body", the " :raw" modifier specifies that MIME decoding should be performed; in the case of "address" and "header" tests, the MIME processing in question is the decoding of MIME encoded-words (see [RFC 2047](#) for the definition of encoded-words). " :text" is the default for "header" and "address" tests on " :comment" and " :display" address parts. " :raw" is the default for other sorts of "address" tests. (7.0.5)
- The proposed "regex" extension adds a " :regex" match type. (6.1)

- The `:resent` and `:noresent` arguments are supported on the [Sieve "redirect" action](#), for controlling whether Sieve "redirect" actions cause addition of Resent-* header lines. (6.3p1)
- The `:resetmailfrom` and `:keepmailfrom` parameters are supported on the [Sieve "redirect" action](#), to control whether or not the original message's envelope From address is used on the redirected message (*vs.* using the Sieve owner's address as the envelope From address for the redirected message). (6.3p1)
- The ["setenvelopefrom" action](#), to override a message's original envelope From address, is available for use in system-level Sieves. (8.0)
- The ["setmtpriority" action](#) is available for system-level Sieves. It accepts a single integer or string argument and sets the current MT-PRIORITY to the argument value. This action is only allowed in system-level Sieves and the argument must be in the -9 to 9 range of valid MT-PRIORITY values. (8.0)
- The ["setnotify" and "setreturn" actions](#) are available in system-level Sieves. (7.0u1)
- The ["setoperation" action](#) is available for system-level Sieves, to set the enqueue operation mode to "submit", "passthrough", "relay", or "default". (8.0)
- The ["setpriority" action](#) is available for system-level Sieves, to set an [effective message processing priority](#). It takes a single string argument which must be one of "non-urgent", "normal", or "urgent". Note that this priority is NOT stored in the message header and only affects processing at this particular stage of message transfer. If multiple "setpriority" actions are specified in different system-level Sieves, the one in the most specific Sieve wins. (7.0u4)
- The ["spamadjust" and "virusset" actions](#) are supported; they tell the MTA how to set the spam or virus score which a standard "spamttest" and "virustest" test, respectively, would then test. (6.0)
- The "strongrandom" function takes as argument an integer value n specifying the number of bytes of (cryptographically strong) random material to return. "strong" is permitted only in system-level Sieves; (attempts to use "strongrandom" from user-levels Sieves are prohibited because it would provide a way for users to drain available entropy from the system random number generator).
- The "transactionlog" action is available for system-level Sieves. It takes a single string argument. All of the "transactionlog" actions in all of the applicable Sieves are concatenated into a single string, which is then available for inclusion in the MTA message transaction log file; see the [log_transactionlog](#) MTA option. (8.0)
- The [translate](#) function is available, to convert a string from one charset to another. (8.0.1)
- The [warn](#) action is available, to cause additional text to be added to the "warn" clause in the [log_filter](#) field of the [MTA message transaction log file](#). (8.0)
- Custom tests may be defined via MTA [FILTER_testname mapping tables](#). (Updates to JES MS 6.2 and updates to JES MS 6.3)
- [Subroutines](#) are supported. (8.0)

Finally, one of the most important of the MTA's extensions to Sieve is the MTA's support of a [hierarchy of Sieve filters](#).

5.1.2.1 Sieve address test

The "address" test is a standard part of the Sieve language. However, MTA support for the "address" test deserves a few special comments.

In regards to "address" tests, note that the MTA supports the ["subaddress" extension](#) (capability name "subaddress") defined in [RFC 3598](#), which makes available `:user` and `:detail` tagged arguments.

New in Messaging Server 7.0.5, the Sieve "address" test supports several enhancements:

- The Sieve "address" test now uses the MTA's heuristic address parser (instead of the strict address parser). This helps "address" tests work even when the overall header contains one or more syntax errors.
- The "address" test supports a private `:aindex` tagged argument. This `:aindex` tag accepts a positive integer as an argument. If `:aindex n` is specified, then only the *n*th address in each header field will be tested. Note that `:aindex` (selecting which address out of multiple addresses) may be combined with `:index` (selecting which header line out of multiple header lines), to cause the test to operate on a single address in a single header line.
- The "address" test supports two new, private, part modifiers, `:display` and `:comment`. The `:display` modifier causes the "address" test to operate on the display-name phrase; note that the display-name phrase is the optional text appearing outside of and specifically in front of the actual address, the actual address being the part enclosed in angles. The `:comment` modifier causes the "address" test to operate on any parenthetical comments that appear after an address.
- The `:raw` and `:text` modifiers defined for the ["body" test](#) may also be used in "address" tests (as well as "header" tests). Similarly to when used with "body", the `:raw` modifier specifies that MIME decoding should be performed; in the case of "address" and "header" tests, the MIME processing in question is the decoding of MIME encoded-words (see [RFC 2047](#) for the definition of encoded-words). `:text` is the default for "header" and "address" tests on `:comment` and `:display` address parts. `:raw` is the default for other sorts of "address" tests.

5.1.2.2 Sieve body extension

New in Messaging Server 7.0u2, the MTA supports the Sieve body extension specified in [RFC 5173 \(Sieve Body Extension\)](#), with the following restrictions:

- The only match types supported are `:contains` and `:is`; while `:matches` and `:regex` are not supported. This is likely to be a permanent restriction due to the possible performance impact of supporting these match-types.
- The only body transforms supported are `:raw` and `:text`; while `:content` is not supported. This restriction is likely to be lifted in a future release.
- [Variable substitutions](#) are not allowed in body test arguments. If they are used an error is likely to occur. This is so that a list of all arguments to body in all scripts can be computed in advance and searched for in a single pass. If this restriction were to be lifted, it would be easy to construct scripts that would require an arbitrary number of passes over the message, which is unacceptable in a server environment. As such, this should be considered to be a permanent restriction. For example, this script will fail:

```
require ["variables", "body"];
set "a" "testing";
if body :contains "${a}" { discard; }
```

- The `:text` body transform operates on all message parts with a text type or a 7bit/8bit encoding. If a charset other than utf-8 is specified on a text part, then that part is converted to utf-8 before being searched.

Note also that new in Messaging Server 7.0.5, `imexpire` supports use of the Sieve body extension.

The availability of the body test is controlled by the `enable_sieve_body` MTA option. A value of 0, the default, disables the extension. A value of 1 enables the extension for use in all Sieves. A value of 2 enables the use of body in system-level Sieves only. Each Sieve script that wishes to use "body" must also declare it using the "body" capability:

```
require "body";
```

5.1.2.3 Sieve copy extension

The MTA supports the Sieve copy extension specified in [RFC 3894 \(Sieve Extension: Copying Without Side Effects\)](#), which adds a `:copy` parameter to the Sieve `fileinto` and `redirect` actions allowing these actions to be used without cancelling the default save-to-inbox Sieve effect. The capability name is "copy":

```
require "copy";
redirect :copy "another-mailbox@domain.com";
# A copy of the message will still be retained/delivered "normally"
```

5.1.2.4 Sieve discard and jettison actions

In addition to the standard "discard" action, the MTA also supports a private "jettison" action (capability name also merely "jettison") which causes a "non-overridable" discard. "jettison" is similar to "discard" in that it causes messages to be silently discarded. The difference between "jettison" and "discard" is that 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 Sieves](#) are involved. For example, a system-level "discard" can be overridden by a user Sieve explicitly specifying "keep", whereas a system-level "jettison" will override anything done by a user Sieve.

Whether the MTA immediately discards a message upon a Sieve "discard" or "jettison" action being applied, *vs.* routing such messages to the `filter_discard` channel (for a short period of retention before being permanently deleted) may be configured via the `filter_discard` and `filter_jettison` MTA options. Configuring use of the `filter_discard` channel allows a system administrator to, if desired, "fetch back" messages a user has very recently, mistakenly, discarded via a Sieve filter. As such, it may be useful either for debugging purposes, or for assisting users prone to setting up overly aggressive discarding via personal Sieve filters. (Technical note: When the MTA routes a discarded message to the `filter_discard` channel, the MTA also sets a [bit in the message envelope](#) that means that subsequent "discard" or "jettison" actions will be ignored. This has no effect for the discarded messages that remain in the `filter_discard` queue until eventually deleted

from disk. However, if instead a "retrieval" procedure is performed on such a message, such as if a system administrator moving a message from the `filter_discard` channel to the `reprocess` channel for subsequent processing, the bit permits delivery to proceed bypassing any "discard" or "jettison" actions.)

When discarded messages are deleted immediately (rather than being routed to the `filter_discard` channel), note that that is logged in [MTA message transaction logging](#) as if the message had been enqueued to the `bitbucket` channel, though in reality no such enqueue is performed and instead the message temporary file is merely deleted from disk.

As of Messaging Server 7.0u4, the "discard" and "jettison" Sieve actions now accept a single, optional string parameter. This parameter value, if specified, is logged as part of the `log_filter` filter result field in the [MTA message transaction log](#), assuming of course that the associated Sieve is the one that determines the disposition of the current message. Note that this argument is nonstandard; however, since the main application is in system-level Sieves, this should not present a portability problem in practice.

Note that "discard" or "jettison" like effects may be obtained via [*_ACCESS mapping table \\$V, \\$v, \\$Z, or \\$z flag](#) use.

Note that application of "jettison" will disable user-level Sieve "vacation" or "notify" actions that might otherwise apply. (Prior to Messaging Server 7.0.5, "jettison" would cancel all "notify" actions; as of Messaging Server 7.0.5, system-level "notify" actions are not cancelled by "jettison".)

5.1.2.5 Sieve date and index extensions

As of Messaging Server 7.0u4, the MTA supports the Sieve date and index extensions defined in [RFC 5260 \(Sieve Email Filtering: Date and Index Extensions\)](#). (Note that prior to Messaging Server 7.0u4, certain parts of these extensions, notably the "currentdate" test and parts of the "date" test, had already been made available.) The "date" test matches dates off header lines; the "currentdate" test matches the current time (the time at which the Sieve script is evaluated); and the "index" extension adds ":index" and ":last" arguments to the "address", "header", and "date" tests. The capability strings are "date" (which enables both "date" and "currentdate" tests) and "index":

```
require ["date", "index"];
```

Some examples of "date" or "currentdate" use:

```
require ["date", "vacation", "relational"];
if anyof(currentdate :is "day" "05",
         currentdate :is "day" "10",
         allof(currentdate :is "weekday" "2",
              currentdate :value "gt" "14",
              currentdate :value "lt" "22"))
{ vacation "I'm working at the hospital today";
  redirect "hospital-address"; }
```

```
require ["date", "relational", "vacation"];
if allof(currentdate :value "ge" "date" "2007-06-30",
         currentdate :value "le" "date" "2007-07-07")
{ vacation :days 7 "I'm away during the first week in July."; }
```

```
require ["variables", "date", "relational"];
set "startDateTime" "2007-01-18T00:00:00Z";
set "endDateTime" "2007-01-19T00:00:00Z";
if allof(currentdate :value "ge" "iso8601" "${startDateTime}",
        currentdate :value "le" "iso8601" "${endDateTime}")
    { redirect "user+wherever@domain.com"; }
```

Note that "currentdate" and "date" by default returns time values in the server local time zone, meaning that [ISO 8601](#) strings such as 2007-01-19T00:00:00-08:00 may be returned. Alternatively, one may use ':zone "+hhmm"' in the test to force shifting (conversion) of the original time zone to the specified time zone prior to performing the test. In contrast, specifying ":originalzone" for a "date" test forces use and retention of the time zone offset originally present.

The [Sieve extlists extension](#), among other things, also adds a ":list" match type to the "date" and "currentdate" tests.

When considering ":index", note that as of Messaging Server 7.0.5 the MTA supports a private ":aindex" tag on [Sieve address tests](#). This ":aindex" tag accepts a positive integer as an argument. If ":aindex n" is specified, then only the nth address in each header field will be tested. Note that ":aindex" (selecting which address out of multiple addresses) may be combined with ":index" (selecting which header line out of multiple header lines), to cause the test to operate on a single address in a single header line.

5.1.2.6 Sieve duplicate extension

As of the 8.0 release, the MTA supports the Sieve duplicate extension specified in [RFC 7352](#). The "duplicate" test is intended to assist in detecting and handling cases of so-called "duplicate" messages such as cases where a user receives both a personally addressed copy as well as a mailing list copy of a message.

The capability string to enable use of the "duplicate" test is "duplicate":

```
require "duplicate";
```

Furthermore, to permit "duplicate" tests, the [max_duplicates](#) and [duplicate_tracking_url](#) MTA options must have, respectively, a positive value and a valid URL value. The [max_duplicates](#) MTA option controls how many "duplicate" tests may be performed in a single Sieve script; the default is 2.

In the MTA's implementation, the MTA maintains a [memcache database](#) recording "recent" duplicate message tracking data: this database is what allows comparing a current message with a prior message to detect a "duplicate". The [duplicate_tracking_url](#) MTA option specifies where this duplicate tracking information should be stored; at present the value must be a [memcache: URL](#) of the form:

```
memcache://host:port/key-prefix
```

If the host isn't specified, it defaults to the value of the [memcache_host](#) MTA option. It is an error for [memcache_host](#) not to be set in this case. If the port isn't specified, it defaults to the value of the [memcache_port](#) MTA option; if that option in turn isn't specified, the default

is 11211, the usual port for memcache servers. `key-prefix`, if specified, is prepended to the keys the duplicate extension sends to the memcache server.

Note that "duplicate" tests are performed during Sieve evaluation, but no memcache updates are performed at the Sieve evaluation stage. It is only after the message has been successfully processed that updates are done.

Also note that duplicate information is implicitly qualified by the [owner of the Sieve](#). In the case of system-level Sieves, this will be the applicable [postmaster address](#), so system-level Sieves operate in shared namespace(s). Note that the `:handle` argument of the "duplicate" test can be used to force system-level Sieves to operate in their own namespace.

The syntax for the "duplicate" test is:

```
duplicate" [":handle" handle-string]
           [":header" header-name-string /
           ":uniqueid" value-string]
           [":seconds" number] [":last"]
```

where the default is to test for previously seen (within a short period of time) values of the Message-id: header line, or previously seen values of another header line instead if `:header` is specified, or previously seen other values constructed as the Sieve chooses per the `:uniqueid` argument string. The time period for which the "seen" values are retained, so the time period within which duplicates may be detected, may be controlled by use of the `:seconds` argument, or if not specified defaults to the value of the [duplicate_timeout_default](#) MTA option.

As of the 8.0 release, warnings that occur during Sieve evaluation such as issues with the duplicate extension (or issues involving the memcache protocol or the [vacation](#) extension), plus any specifically specified warning text specified via the [Sieve "warn" action](#) will result in a "warn" clause in the [log_filter](#) field of [MTA message transaction log entries](#).

5.1.2.7 Sieve editheader extension

The MTA has supported the `addheader` action (prior to its standardization) since circa JES MS 6.1, and the standard `deleteheader` action and proposed `replaceheader` action since JES MS 6.3. The standard capability string in order to use an `addheader` or `deleteheader` action, as defined in [RFC 5293 \(Sieve Email Filtering: Editheader Extension\)](#), or a `replaceheader` action, as defined in [draft-degener-sieve-editheader-00](#), is "editheader", although note that the MTA does not enforce this for the `addheader` action (`addheader` may be used without a "require" clause):

```
require "editheader";
```

The MTA has a configurable limit on how many `addheader` actions will be permitted in a single Sieve script, [max_addheaders](#). The default is 10. As of the 8.0 release, this limit only applies to user-level Sieves.

When specifying a header label in an `addheader` action, note that the header label length is limited to 256 characters, and may not contain any eight bit characters (characters above ASCII position 126) nor control characters (characters below ASCII position 33) as well as not containing the colon character, `:`.

Note that it may often be useful to make use of the [Sieve variables extension](#) along with `editheader`, and perhaps especially in conjunction with the `replaceheader` action. This is illustrated in the following example in which a site's broken DMARC usage, which could break mailing lists *for innocent other members of the list* is ameliorated by forcibly modifying the (broken domain's) addresses so as not to trigger bounce messages for messages from this broken domain to other list recipients thereby causing the innocent list members to be removed from mailing lists.

```
require ["editheader", "variables"];
if address :domain :is "From" "dmarcbrokenusage.domain.com" {
# dmarcbrokenusage.domain.com addresses that include phrase and/or comment:
  replaceheader :newvalue "${1}<${2}@dmarcbrokenusage.domain.com.invalid>${3}"
    :matches "From" "**<*@dmarcbrokenusage.domain.com>*";
# Simple dmarcbrokenusage.domain.com addresses:
  replaceheader :newvalue "${1}@dmarcbrokenusage.domain.com.invalid"
    :matches "From" "**@dmarcbrokenusage.domain.com";
  addprefix text:
Due to dmarcbrokenusage.domain.com's broken use of DMARC, the From: address
in this message has been replaced by <original-address>.invalid.

To reply to the original sender of this message, remove the .invalid
from the end of the domain.

.
}
```

5.1.2.8 Sieve envelope extension

The capability string in order to use an envelope test, as defined in [RFC 5228](#) (or originally in [RFC 3028](#)), is "envelope":

```
require "envelope";
```

The "envelope" test has been further extended by additional RFCs to allow access to additional envelope fields, including the "envelope-dsn" extension defined in [RFC 6009](#) for access to the information provided by the DSN SMTP extension, and the "envelope-auth" extension, for access to the SMTP AUTH value; while other RFCs extend "envelope" as well as other Sieve operations, including the "subaddress", "extlists", and "relational" extensions which among other effects also supplement the range or types of allowed "envelope" tests. And the MTA's private [conversion tag mechanism](#) can also be accessed from "envelope" tests using the private "conversiontag" part.

In order to use such supplementary "envelope" parts, the additional extension, as well as "envelope" itself, must be listed in a "require" action (all except for "conversiontag" which does not need a "require" action); *e.g.*:

```
require ["envelope", "envelope-dsn"];
require ["envelope", "envelope-auth"];
require ["envelope", "subaddress"];
require ["envelope", "extlists"];
```

```
require ["envelope", "relational"];
```

The "envelope" test takes a string or list argument.

In addition to supporting the standard envelope test arguments specified in [RFC 5228](#) and the other extensions mentioned above, the envelope test supports a `conversiontag` argument. This test checks the [current list of tags](#) associated with the current recipient, one at a time. Note that the `:count` modifier (from the ["relational" extension](#)), if specified, allows checking of the number of active conversion tags. This type of envelope test is restricted to system-level Sieves. Also note that this test only "sees" the set of conversion tags that were present prior to Sieve processing; the effects of ["setconversiontag"](#) and ["addconversiontag"](#) actions are not visible.

When using an "envelope" test from a non-channel application or utility such as [imexpire](#), note that the [External filtering context MTA options](#) may be relevant.

5.1.2.9 Sieve environment extension

New in Messaging Server 7.0-3.01, the Sieve environment extension specified in [RFC 5183 \(Sieve Email Filtering: Environment Extension\)](#) has been implemented. All of the items defined in the RFC are provided, namely: `domain`, `host`, `location`, `name`, `phase`, `remote-host`, `remote-ip`, `version`. Additionally, the `vnd.sun.source-channel` item returns the name of the current source channel and the `vnd.sun.destination-channel` item returns the name of the current destination channel.

New in Messaging Server 7.3-0.01, the `vnd.sun.autoreply-internal` item returns TRUE or FALSE according to whether the autoreply criteria have been met for use of the ["internal" autoreply response text](#). Also new in Messaging Server 7.3-0.01, the MTA supports two new Sieve environment items, `vnd.sun.authenticated-sender-address` and `vnd.sun.authenticated-sender-id`. The former provides access to the sender address that's associated with the authentication state for the SMTP session. The latter provides similar access to the user identity.

New in 7.0.5, the MTA supports the new Sieve environment item `vnd.oracle.last-verdict`. When the [Sieves associated with a recipient are evaluated in order](#), each evaluation that performs an explicit handling action sets this item as it finishes so the next Sieve in the sequence can check it. A Sieve script that doesn't perform an explicit handling action will leave this item unchanged. Possible values that can be set are:

- `refuse`
- `reject`
- `ereject`
- `jettison`
- `fileinto`
- `redirect`
- `keep`
- `discard`

Note that testing the `vnd.oracle.last-verdict` environment item makes the Sieve script recipient-specific in the same fashion an envelope "To" test does, and will result in this and subsequent Sieves being reevaluated for every recipient. Although any script can test this item, it is intended for use when other applications and utilities such as [imexpire](#) ask the MTA to perform antispam and antivirus checks.

New in 8.0, the MTA supports a new Sieve environment item, `vnd.oracle.mt-priority`. This item returns the current [MT-PRIORITY value](#) as a string.

New in 8.0, the MTA supports a new Sieve environment item, `vnd.oracle.operation-type`. This item returns the current [type of enqueue operation](#) that is underway. The possible values are "DEFAULT", "SUBMIT", "RELAY", and "PASSTHROUGH".

Arbitrary, custom environment items may be set via the [FROM_ACCESS mapping table](#) or any of the [recipient *_ACCESS mapping tables](#) using the `$(E)` flag. The value of such a custom environment item may subsequently be tested in a Sieve script. For instance, the following `FROM_ACCESS` entry defines, for messages coming in via the SMTP port from the `tcp_local` channel (the Internet), the custom environment item "heloname", giving it the value "yes" of the name the SMTP client claimed on the HELO/EHLO line:

```
FROM_ACCESS
```

```
TCP|*|25|*|SMTP|*/*|MAIL|tcp_local|*          $$Eheloname|$(E)
```

A Sieve script may then test this custom "heloname" environment item:

```
require ["environment","fileinto"];
# First, check if heloname item is set:
if environment :contains "heloname" "" {
# If heloname item IS set, then check its value:
    if environment "heloname" :is "Bogus Name" {fileinto "bogus"; }
}
```

5.1.2.10 Sieve ereject and reject and refuse extensions

The original Sieve specification, [RFC 3028](#), defined the optional "reject" extension and action as being required to result in a [Message Disposition Notification](#). [RFC 5429](#) redefined "reject" to allow it to refuse messages during the SMTP transaction (rather than accepting messages and generating back a separate MDN), and defined the "ereject" action to require messages to be refused during the SMTP transaction. The MTA also supports the nonstandard "refuse" action, capability name "refuse", which was an earlier draft approach similar to "ereject": note that "refuse" attempts to do an SMTP level rejection but falls back to an MDN when SMTP level rejection is not feasible (as for instance in the case of a multi-recipient message where not all recipients are to be rejected), whereas "ereject" will, if SMTP rejection is not feasible, discard messages with forged return-path or fall back to a DSN. The capability name for "reject" with its original behavior (MDN generation) is "reject"; the capability name for the "ereject" action and for the updated behavior from the "reject" action is "ereject".

Note that "ereject", as per its design from [RFC 5429](#), is only intended to be made available on ingress MTAs capable of returning an SMTP level error directly to remote systems in other administrative domains; the [enable_sieve_ereject](#) MTA option which enables "ereject" availability is on by default, but may (and should) be disabled on "internal" MTA hosts.

When "refuse" support was first added (JES MS 6.1), "refuse" was only supported in system-level Sieve scripts. That restriction was removed for JES MS 6.2. Also, prior to Messaging Server 7.0, a "refuse" for any recipient caused an SMTP-level "refuse" for all recipients; as of Messaging Server 7.0, instead "refuse" performs an SMTP-level rejection only

if "refuse" applied to all recipients and otherwise falls back to generating an MDN regarding the "refuse" recipient(s). Also as of Messaging Server 7.0, a "refuse" in a more general (*e.g.*, system-level) Sieve will override actions taken by more specific (*e.g.*, user-level) Sieves.

Each of "ereject", "reject", and "refuse" takes a single string argument specifying error text to include in the SMTP error or notification message.

Note that "ereject", "reject", and "refuse" cannot be combined with anything except "discard", "capture", "vacation", or "hold".

Note that any Sieve "notify" actions in a user-level Sieve script will be automatically cancelled when the overall Sieve verdict is "ereject", "reject", or "refuse" (or "jettison"); but "notify" action in a system-level Sieve script will, as of Messaging Server 7.0.5, be honored despite the overall verdict, thus making it possible to use "notify" for some limited administrative auditing purposes.

New in 8.0, the "reject", "ereject", and "refuse" actions will parse any extended SMTP error code (*e.g.*, "5.7.2") that appears at the beginning of the action's string argument in any system-level Sieve script, and use it in preference to the default 5.7.1 extended SMTP error code. This feature is not available to user-level Sieve scripts.

5.1.2.11 Sieve external lists

[RFC 6134 \(Sieve Extension: Externally Stored Lists\)](#) defines a Sieve extension for external lists, EXTLISTS, which is intended for cases where a Sieve script would like to consult an externally-stored list: this might be data in an LDAP directory, a flat file containing a list of something, a database, a personal addressbook, *etc.* In order to use a Sieve external list, a Sieve script must, per Sieve syntax, declare that it will use this extension via

```
require "extlists";
```

The MTA's implementation of EXTLISTS operates as follows. Sieve scripts may use the ":list" argument in a "redirect" action or in tests such as "address", "envelope", "header", "string", "spamtest", or "virustest", *etc.*, or in the test component of the "deleteheader" or "replaceheader" actions, to indicate a wish to access an "external list".

The MTA then makes use of the SIEVE_EXTLISTS mapping table to determine the meaning and contents of the referenced external list.

When an external list is referenced in a Sieve script, via a ":list" argument, *e.g.*,

```
if address :list "from" "pab" { keep; }
```

or

```
redirect :list "friends";
```

the MTA uses the list name (as well as other data) to construct a probe into its SIEVE_EXTLISTS mapping table. In the case of tests, the probe syntax is:

```
test-name | sieve-owner | spare_4-value | spare_5-value | spare_6-value | list-name | argument-value
```

Here test-name can be any of hasflags, address, envelope, spamtest, virustest, header, string, environment, currentdate, date, replaceheader, deleteheader.

The sieve-owner is normally the (canonical address of the) "owner" of the Sieve---so the

user's address for user-level Sieves, or normally the [postmaster address](#) for system-level Sieves; or the field can be blank if the Sieve has no owner address. The `spare_4-value`, `spare_5-value`, and `spare_6-value` fields contain the values of those LDAP attributes named by the MTA's `ldap_spare_4`, `ldap_spare_5`, and `ldap_spare_6` MTA options associated with the current envelope recipient; these `ldap_spare_*` options have no defaults, so by default no values appear here. Furthermore, even if `ldap_spare_*` MTA options are defined, the `spare_*` fields will be blank if the current recipient address was not obtained from an LDAP entry. (And prior to Messaging Server 7.2-0.01, the `ldap_spare_*` fields would be blank for system Sieve probes; as of Messaging Server 7.2-0.01, the current recipient address's LDAP attribute values are used even for system Sieve probes.) The intention is that such `ldap_spare_*` MTA options (and corresponding attributes) may be defined and used to store, or a per-user basis, information about how to construct appropriate access URLs for different "types" of external lists: for instance, one of these attributes might store a value which is a sort of template for constructing lookups of lists in that user's personal addressbook, while another of these attributes might store a value which is a sort of template for constructing lookups of lists in that user's calendar (so a CardDAV lookup template). The `list-name` is the argument to the `:list` parameter; for instance, in the examples above "pab" and "friends", respectively. (In fact, consulting multiple external lists is supported, in which case the multiple list names are present in the probe, separated with the vertical bar character.) Finally, the `argument-value` is whatever string from the message corresponds to the Sieve test.

For "redirect" actions, the syntax is:

```
redirect | sieve-owner | spare_4-value | spare_5-value | spare_6-value | list-name
```

where note that there is no final `argument-value`. (The Sieve external lists draft has another, alternate construct for `redirect :list`, where the argument assumed above to be a list name is instead a URL to which to do the redirection. While the MTA is capable of supporting such usage, enabling it by matching and returning the argument blindly would be extremely dangerous and is *not* recommended. Recommended configuration and usage is instead to have the `SIEVE_EXTLISTS` mapping table: (1) expect and match only when a list name is the argument, and (2) construct and return a sensible URL based upon that list name as well as the other fields of the probe.)

The `S` flag will be set if the Sieve is a system Sieve, and may be tested in the template using the usual `:$` (flag set) or `;!S` (flag clear) flag test syntax.

In the case of tests, the mapping table template (right hand side) should return the actual matching value(s) (that is, the expanded list). In the case of a `redirect` action, the template should return a URL pointing to the desired list of addresses. The mapping template must set the `$Y` flag for the result to be used. In addition, if a Sieve test or "redirect" making use of a Sieve external list employs any of the `spare_*` fields, then the mapping template must also set the `$*` flag. `$*` tells the Sieve machinery that the test is recipient-specific and the script must be reevaluated for each recipient. Failure to set `$*` can lead to botched test results for multirecipient messages.

An MTA-specific extension to the Sieve external lists draft is that the MTA will also potentially return properties associated with list entries. If [Sieve variables](#) are enabled, then the use of `:list` sets variables in a fashion similar to `:matches`: that is, `#{0}` is set to whatever was found on the list, `#{1}` is set to the first property value associated with the list entry, `#{2}` is the second property value, *etc.* When variables are enabled, the result of the mapping consists of properties separated by vertical bars.

So for instance, suppose a site has the `ldap_spare_5` MTA option defined naming an LDAP attribute in which the users store the leading portion of the LDAP URL for where each user's own, personal, PAB information is stored; for instance, suppose that in the user attribute `psroot` is each user's PAB DN location information in a form of:

```
ldap:///user's-pab-dn
```

Then suppose further that the site wants users to be able to access their personal PAB information for `:list` use in their personal Sieve filters. In particular, the site wants the list named "pab", if used in an address or envelope test, to mean "any address found in the user's own PAB", and for any other, more specifically named list `list-name`, to mean the contact entries in a user's own PAB that are in the `list-name` PAB group; that is, the contact entries marked with "memberOfPiGroup=list-id" where the `list-id` was defined in a group entry in the user's PAB (an entry with `objectClass=PiTypeGroup` and with `displayName=list-name` and `piEntryID=list-id`). Also suppose that the site uses the `piEmail1` attribute to store the address most suitable for using to send to users. Then the site might want an `ldap_spare_5` setting of

```
# msconfig set ldap_spare_5 psroot
```

where in legacy configuration mode, the setting would be made in the MTA option file as

```
LDAP_SPARE_5=psroot
```

And the might would also want a `SIEVE_EXTLISTS` mapping table including entries such as:

```
SIEVE_EXTLISTS

! Check for the special case of testing whether an 'address' is merely
! present in the user's PAB somewhere -- the test of external list "pab":
!
! test-name|sieve-owner|spare_4|spare_5|spare_6|pab|address-from-message
!
! address|*|*|*|*|pab|* \
! $]pab$1?piEmail1?sub?(|(piEmail1=$4)(piEmail2=$4)(piEmail3=$4))[$Y
! envelope|*|*|*|*|pab|* \
! $]pab$1?piEmail1?sub?(|(piEmail1=$4)(piEmail2=$4)(piEmail3=$4))[$Y
! header|*|*|*|*|pab|* \
! $]pab$1?piEmail1?sub?(|(piEmail1=$4)(piEmail2=$4)(piEmail3=$4))[$Y
!
! Note that no entry to allow redirect to the pseudo-list "pab" is included
! above: this is intentional as making it "too easy" for users to resend to
! all their contacts seems unwise. Instead, redirect is enabled below merely
! for specifically named and defined PAB groups.
!
! Now check for named groups (named external lists); that is, for a named
! group, find the piEntryID for that group.
!
! test-name|sieve-owner|spare_4|spare_5|spare_6|group|address-from-message
!
! address|*|*|*|*|*|* \
! $CGROUP|address|$2|$5|$]pab$1?piEntryID?sub?(&(objectClass=piTypeGroup)(displayName=$4)) [
! envelope|*|*|*|*|*|* \
! $CGROUP|envelope|$2|$5|$]pab$1?piEntryID?sub?(&(objectClass=piTypeGroup)(displayName=$4)) [
! header|*|*|*|*|*|* \
! $CGROUP|header|$2|$5|$]pab$1?piEntryID?sub?(&(objectClass=piTypeGroup)(displayName=$4)) [
! redirect|*|*|*|*|*|* \
```

```

$CGROUP|redirect|$2|$]pab$1?piEntryID?sub?(&(objectClass=piTypeGroup)(displayName=$4)) [
!
! Now find contact entries that have the correct piEntryID, probing with
!
! GROUP|test-name|spare_5|address-from-message|piEntryID
!
GROUP|address|*|*|%* \
$]pab$0?piEmail1?sub?(&(memberOfPiGroup=$2$3)(|(piEmail1=$1)(piEmail2=$1)(piEmail3=$1))[$Y
GROUP|envelope|*|*|%* \
$]pab$0?piEmail1?sub?(&(memberOfPiGroup=$2$3)(|(piEmail1=$1)(piEmail2=$1)(piEmail3=$1))[$Y
GROUP|header|*|*|%* \
$]pab$0?piEmail1?sub?(&(memberOfPiGroup=$2$3)(|(piEmail1=$1)(piEmail2=$1)(piEmail3=$1))[$Y
!
! Note redirect case needs to return PAB URL itself, rather than lookup result.
! It is assumed that for all lists, the most appropriate user addresses to use
! for redirection are stored in the users' piEmail1 attribute.
!
GROUP|redirect|*|*|%* pab$0?piEmail1?sub?(memberOfPiGroup=$1$2)$Y

```

Note here how the user's initial portion of an LDAP URL value, stored in the attribute named by `ldap_spare_5`, is converted by the `SIEVE_EXTLISTS` mapping table into the MTA's [pabldap: URL format](#) (the format that tells the MTA that this is an LDAP URL for querying the PAB directory---the LDAP directory that the MTA locates via its `ldap_pab_*` options). For the special case of an "address" or "envelope" test against the so-called list named merely "pab", the user's own entire PAB is searched, looking for entries where the test argument matches the value of any of the attributes `piEmail1`, `piEmail2`, or `piEmail3` in a PAB entry. For other named lists/groups, the list/group name is used to attempt to locate a corresponding entry (with `objectClass` of `piGroupType` and a `displayName` matching the specified group/list name), and when found, retrieve the `piEntryID` value that indicates that group/list. Then a second lookup is performed, to find those contact entries that have a `memberOfPiGroup` value for the group/list in question. In the case of an "address" or "envelope" test against named lists, the user's PAB is searched looking for entries where both the list ID is present in the entry in a `memberOfPiGroup` attribute, and the entry's `piEmail1` value matches the test argument. Just in case a user configures a "header" test for address matching purposes (though note that this is poor user practice, as "address" or "envelope" tests are more appropriate for such purposes), "header" tests are set up similarly, though including matching against alias values (`piEmail2` and `piEmail3` in addressbook attributes, analogous to the MTA's usual `mailAlternateAddress` and `mailEquivalentAddress` attributes) as well as against the canonical `piEmail1` (analogous to MTA's `mail` attribute) value. And in the case of a "redirect" action, the mapping returns not the actual addresses, but rather a "pabldap:" URL specifying where to find the appropriate addresses to which to redirect; this URL will be the new address (the address to which to redirect) enqueued to the reprocess channel, which in turn will perform the actual list expansion (expand that URL into the addresses it specifies). Note that the [max_redirect_addresses MTA option](#) limits how many addresses will actually be used from such a list; additional addresses will be ignored.

With Sieve access to user personal PAB's set up as above, then users can make use of the addresses in their PAB in various ways in their Sieve scripts.

As a simple example, suppose that a system-level Sieve would do a

```
discard;
```

on a message for whatever reason: perhaps because a spam/virus filter package callout returned a verdict suggesting that a message is spam; perhaps due to the source-IP being suspect; whatever. But if a user wishes to keep any message purporting to come from one

of their known correspondents, then that user might use a personal Sieve filter with explicit `keep` and `fileinto` actions to override the system-level `discard`, e.g.,

```
require ["envelope","extlists"];
...other-actions-such-as-list-fileintos...
if envelope :list "from" "pab" { keep; }
```

Or for another example, suppose that a system-level Sieve filter (perhaps configured as part of a spam/virus filter package callout so configured via a `spamfilter_n_action_m` MTA option value) has a Sieve effect of:

```
require ["spamtest","relational"];
if spamtest :value "ge" "200" { discard; }
```

Then a user might use a Sieve test and action such as:

```
require ["envelope","extlists"];
if envelope :list "from" "pab" { spamadjust "-10000"; }
```

In this example above, the user adjusts the `spamtest` value downward (drastically) for any sender address found in the user's own PAB. With such a drastically lowered `spamtest` value, the message will likely be safe from any system-level Sieve `spamtest` that might otherwise choose to discard or reject the message. (And the user's Sieve can continue on to do any further `fileinto` or other operations that may seem desirable to the user. Leaving open the potential for the user's own Sieve to perform further actions, such as `vacation` or `fileinto` or `redirect`, is a reason why a user might prefer to do a `spamadjust` rather than an explicit `keep` for known senders.)

Or the user's Sieve might do more subtle adjusting and testing:

```
require ["envelope","extlists","fileinto","spamtest","relational"];
/* First, lower the spam score for senders in my PAB */
if envelope :list "from" "pab" { spamadjust "-100"; }
/* But better check open-list postings -- that list gets
   postings with forged From addresses.  If the spam score
   if over 200 even after any adjustment for being
   sent by a recognized contact, discard the message. */
if allof (header :contains ["To","Cc","Bcc"] "open-list@domain.com",
         spamtest :value "ge" "200") {
    discard;
    stop;
}
/* File messages to or from my buddies in my "softball" PAB list
   or with softball in the subject, to my softball folder */
if anyof (envelope :list ["from","to"] "softball",
         header :contains "Subject" "softball") { fileinto "softball"; }
/* Discard mildly spammy messages (mild after adjustment for known
   senders, above) that don't show me on a recipient header
   line */
if allof (not header :contains
         ["To","Cc","Bcc","Resent-to","Resent-Cc","Resent-Bcc"]
```

```
        "my-own-address",
        spamtest :value "ge" "50") {
discard;
stop;
}
/* Keep messages from known (in my PAB) correspondents */
if envelope :list "from" "pab" { keep; }
```

Comparing date with list of holidays

For another example, suppose that a site keeps a list of site-wide holiday days stored in the MTA's general database, in general database entries of the form:

```
HOLIDAY|yyyy-mm-dd      Yes
```

(where for purposes of this example, it is the mere existence of an entry that matters, not the details of what its right hand side translation value---here Yes---may be), and that the site wishes users to be able to perform `currentdate` and `date` tests against that list using the special list named `holiday`. Then the site might use:

```
SIEVE_EXTLISTS
```

```
currentdate|*|*|*|*|holiday|*    ${HOLIDAY|$4}$Y
date|*|*|*|*|holiday|*          ${HOLIDAY|$4}$Y
```

This would then allow users to use tests such as

```
require ["date","extlists"];
if currentdate :list "date" "holiday" { redirect "mobile-address"; }
```

A more sophisticated use would be to store a label specifying the type of holiday (*e.g.*, "National", "Municipal", "Religious", "Corporate", *etc.*) as the right hand side of each general database entry. Then Sieve scripts could check that returned value as a property (using the [Sieve variables extension](#) and checking the returned property value via `#{1}`) and perform additional decision making based on the type of holiday.

5.1.2.11.1 Example Sieve external lists with properties

The MTA supports a private feature of [Sieve external lists](#), whereby external lists can return properties associated with list entries. This can be a powerful additional tool. This section presents two examples below, both variants on "capturing" copies of particular messages passing through the MTA.

Capturing a user's "external" messages

Suppose that you wish to capture copies of certain users' Internet correspondence, without bothering to capture copies of those users' internal correspondence (meaning that direct use of an `ldap_capture` LDAP attribute would capture unneeded messages), and that you'd like to keep track of which users are in this category in LDAP, rather than hard-coding such a list directly into a Sieve script. One approach for doing this would be to use channel-level source and destination Sieve scripts on the `tcp_local_channel` (which is the channel handling messages coming in from, or going to, the Internet), where such Sieve scripts make use of an

external list to check LDAP to determine which users' messages are eligible for [capture](#). Using the properties feature of the MTA's Sieve external lists implementation, the external list will also return the capturer address to use (the address to which to send the captured message copies). The components of such an approach are:

1. Add some user-level LDAP attribute to the schema (or disable schema checking) and set that attribute on the users for whom you want capture, with a value which is the address to which to send the captured message copies. (Note that typically such an attribute should have [ACIs](#) so that users themselves can't even see the attribute, let alone change its value.) This example will assume there is an attribute named `mailCaptureInternet` for this purpose. (Note that if you already have `ldap_capture` defined and pointing to the name of some LDAP attribute used for unconditional capture, then you probably don't want to use the same attribute for this "conditional" capture, as that would merely result in an additional capture copy in the "conditional" cases. Instead you want a different LDAP attribute, which will only be consulted and have an effect in this special case.)
2. Set the `ldap_spare_4` MTA option to the name of this "conditional capture" attribute; in unified configuration:

```
msconfig> set mta.ldap_spare_4 "mailCaptureInternet"
```

or in legacy MTA configuration mode, set in the `option.dat` file:

```
LDAP_SPARE_4=mailCaptureInternet
```

Pointing `ldap_spare_4` at this attribute means that the attribute's value will be included in probes of the [SIEVE_EXTLISTS mapping table](#), which will turn out to be convenient.

3. Define Sieve external lists named "capture-to" and "capture-from" via a [SIEVE_EXTLISTS mapping table](#) as follows. (In legacy configuration mode, this [SIEVE_EXTLISTS mapping table](#) should be placed in the MTA mappings file; in Unified Configuration mode, the mapping table can be created by editing from within the `msconfig` utility.)

```
SIEVE_EXTLISTS
```

```
! Define an external list named "capture-to" for use in "envelope" tests of
! the To address.  Because the LDAP_SPARE_4 field of the pattern has a
! match pattern of %*, a probe will match this entry only when the envelope
! To recipient being tested has a non-empty mailCaptureInternet value:
!
!   envelope|*|*|*|*|capture-to|*   $Y*$1$2
!
! When the probe matched, the test succeeds ($Y) and the entry returns
! <mailCaptureInternet-value> for the matched address as the first (indeed
! only) property, so it will be accessible via Sieve ${1} variable.
! Note that because this is a recipient-specific test, making use of the
! LDAP_SPARE_4 value, the entry includes $* in the template.
!
! Now define an external list named "capture-from" for use in "envelope" tests
! of the From address.  Because the Sieve language is oriented towards
! performing actions on behalf of message recipients, obtaining information
! from LDAP regarding the message sender (envelope From) requires some
! additional, explicit LDAP lookups (more than is required for the "capture-to"
! external list case).
! First, get the base DN for the user entries in the domain of the From
! address and rebuild a new probe:
!
```

```

envelope|*|*|*|capture-from|* @* $CBASEDN|FROM|$4@$5|$}$5,_base_dn_{
!
! If the envelope From was that of a user in one of "our" domains, then
! the $}<domain-name>,_base_dn_{ lookup should succeed, so the entry
! succeeded and the probe is now:
! BASEDN|FROM|<from-address>|<basedn-of-from-domain>
!
BASEDN|FROM|*|* \
$C$]ldap:///!$!mailCaptureInternet?sub?(&( |(mail=$0)(mailEquivalentAddress=$0))(mailCaptureInternet=*))[$Y
!
! When this probe matched and the LDAP lookup succeeds, then the test
! succeeds ($Y) and the entry returns <mailCaptureInternet-value>
! as a first property (so accessible via Sieve ${1} variable), thus the
! capture attribute value for that matched address is available.

```

4. On the [tcp_local channel](#) (and any other dedicated-to-Internet-correspondence channel(s)), use a [sourcefilter](#) Sieve along the lines of:

```

require ["envelope", "extlists", "variables"];
if envelope :list "to" "capture-to" { capture "${1}"; }

```

and a [destinationfilter](#) Sieve along the lines of:

```

require ["envelope", "extlists", "variables"];
if envelope :list "from" "capture-from" { capture "${1}"; }

```

Note that this example used the same LDAP attribute `mailCaptureInternet` to determine capture for both incoming and outgoing directions. (The incoming, "capture-to", list took advantage of setting `ldap_spare_4` to conveniently fetch the value of this attribute for the recipient; for the outgoing, "capture-from", list, two separate, explicitly configured LDAP lookups were required to first locate where in the directory to search, and second fetch the actual attribute value.) But separate attributes could be used, if different criteria were desired for incoming *vs.* outgoing. Also, in this example the Sieve external list itself simply checks the attribute value---and the fact that the capture is (intended) for Internet correspondence is incorporated by virtue of the Sieve filters being placed on the Internet correspondence channel (`tcp_local`). More complicated Sieve filter tests combined with this external list consultation could further refine which messages are captured; see for instance, the additional, "attachment type" testing shown in the example below. Or use of a Sieve filter consulting these external lists on different MTA channels could completely alter which messages get captured.

5.1.2.11.2 Testing Sieve external lists

As usual for Sieve filters, the `imsimta test -expression utility` is one way to do some checking and testing on Sieve external lists. And it may be worth making special note that use of the utility with the switch `-debug=3` will show some details of operation of the [SIEVE_EXTLISTS mapping table](#) lookup; this may be especially useful for debugging or confirming correct configuration of a `SIEVE_EXTLISTS` mapping table.

5.1.2.12 Sieve fileinto action

[RFC 5228 \(Sieve\)](#) defines the "fileinto" action as optional since though quite desirable, it may not be possible in some environments. In order to use "fileinto", a Sieve script must, per Sieve syntax, declare that it will use it via

```
require "fileinto";
```

Note that "fileinto" is not supported for domain Sieves. And the number of "fileinto" actions that may be performed by a Sieve script is limited (as of the 8.0 release, this limit only applies to user-level Sieves) by the `max_fileintos` MTA option (default value 10).

To implement a Sieve script's "fileinto" action, the MTA's behavior is controlled by the `fileinto` channel option: that channel option is normally configured to insert the folder-name specified by the Sieve script's "fileinto" argument into the recipient address in the form of a subaddress. Next, the MTA must pass along to the Message Store the decision of whether to "trust" the subaddress for folder delivery purposes; relevant channel options are `deliveryflags` and `flagtransfer`. Note that even if a Sieve script appears to perform a "fileinto" action, the actual delivery-into-a-folder requires that proper configuration have been performed to properly implement the transfer to the Message Store of the desired "fileinto" effect.

The `copy` extension adds a `:copy` tag to "fileinto" (so that the "fileinto" does not, as would be normal, cancel the Sieve "implicit keep"). The `imap4flags` extension adds, among other features, a `:flags` tagged argument (to specify IMAP flags to set on the message as it is delivered).

Note that users are permitted to "fileinto" their *own* folders; in contrast, delivery to another user (or to a desired folder belonging to another user) is *not* a "fileinto" effect but rather requires a `redirect` action. There is one exception to this, and that is the case (such as in cases of `head of household Sieve filters`) where the owner of a Sieve differs from the user on whose behalf the Sieve is being applied; the MTA's private `:owner` tag specifies that the folder named is that of the owner of the Sieve filter, rather than of the user for whom the Sieve is being applied.

5.1.2.13 Sieve ihave extension

As of Messaging Server 7.0, the MTA supports the Sieve `ihave` extension described in [RFC 5463 \(Sieve Email Filtering: Ihave Extension\)](#); the capability name is "ihave". This includes the "ihave" test and the "error" control structure.

The `ihave` test takes as argument a list of capability names and returns `true` if all the listed capabilities are available to the Sieve script. Thus this permits a Sieve script to be coded in such a way as to be flexible regarding what `extensions` it attempts to use, and also potentially to be portable (run in different environments). In particular, when a capability's availability has been confirmed via a successful `ihave` test, then that extension becomes available throughout the entire Sieve script, as if it had been listed in a `require` action. The `error` control structure may be used when it is desired to exit with a runtime error if an `ihave` test fails (a capability is not available).

5.1.2.14 Sieve imap4flags extension

As of JES MS 6.3p1, the MTA supports the Sieve `imap4flags` extension of [RFC 5232](#); the capability name is "imap4flags". This includes the `addflag`, `setflag`, and `removeflag` actions, and the `hasflag` test, as well as the `:flags` argument for the `keep` and `fileinto` actions.

Note that as IMAP system flags always begin with a backslash character, `\`, and as backslash is the quoting character in Sieve, when specifying such an IMAP system flag, the backslash in the flag name must itself be quoted with another backslash, *e.g.*:

```
require "imap4flags";
keep :flags "\\Flagged";
```

An example of setting a user IMAP flag is:

```
require "imap4flags";
if header :contains "Disposition-Notification-To" "**@domain.com" {
    addflag "$MDNRequired";
}
```

Note that as of Messaging Server 7p24 and 7.0.5, `imexpire` supports expiring messages based on user flags.

5.1.2.15 Sieve mime extension

As of Messaging Server 7.0u1, the MTA supports the Sieve mime extension of [RFC 5703 \(Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure\)](#); the capability name is "mime". (The MTA does not, however, support either "replace" or "enclose", also described in [RFC 5703](#); for replacement sorts of effects, see instead the MTA's [charset conversion](#) facility or the [conversion channel](#), and for enclosure sorts of effects, see instead the MTA's message capture facilities.)

New in the 8.0 release is support for the "foreverypart" Sieve extension (from [RFC 5703](#)); the capability name is "foreverypart". In addition to the "break" control command, the MTA's implementation also supports the nonstandard "continue" control command:

```
continue [:name string]
```

"continue" has the expected semantics: control is passed to the bottom of the "foreverypart" loop.

Note that the MTA also supports a nonstandard "loop" extension, discussed under [Sieve loop extension](#). It is wise to stick with use of "foreverypart" when it suffices for a purpose, but the "loop" construct does offer another alternative for more complex loop-based processing.

New in the 8.0 release is support for the "extracttext" Sieve extension (from [RFC 5703](#)). Note that since the MTA's [Sieve support is implemented as an overlay on top of an underlying language interpreter](#), the use of "extracttext" outside of a "foreverypart" is not detected as an error at compile time. Additionally, "extracttext" is only supported on leaf parts: it cannot be used on multipart and message/rfc822 parts.

As of the 8.0 release, the behavior of the Sieve "size" test inside of "foreverypart" loops has been changed. Previously "size" operated on the message as a whole no matter what the context; now it operates on the current part only. Note that only decoded part data is considered; part headers are not included in the size calculation. Also note that the size of non-leaf (message and multipart) parts is currently zero; this may or may not be changed in the future.

This nonstandard extension to the Sieve "size" test is mainly intended to be used to implement attachment size checks. However, since the "size" test can also be used as function call (in which case it returns the size in octets), this can also be used in conjunction with

"foreverypart" to build message manifests for insertion into header fields or logging with the ["transactionlog" action](#).

For instance, one use of "foreverypart" would be to scan the parts of a message to build a so-called "manifest" of the message, where the [addprefix extension](#) could be used to add the manifest to the first text part of the message:

```
require ["variables","mime","foreverypart"];
addprefix "Manifest:";
foreverypart {
  if not anyof (header :mime :type :is "Content-type" "multipart",
               header :mime :type :is "Content-type" "message")
  {
    if header :mime :contenttype :matches "Content-type" "*" {
      addprefix "Part " + "${0}" + ", Size " + size + " characters of content";
    }
  }
}
addprefix "Total size: " + size + " characters including header";
addprefix "End of manifest.\r\n\r\n";
```

In a system-level Sieve (so that `addconversiontag` may be used), the following Sieve script will add a conversion tag incorporating the message's content size as part of the conversion tag:

```
require ["variables","foreverypart"];
counter = 0;
foreverypart { counter = counter + size; };
addconversiontag "size" . counter;
```

As of the 8.0 release, when user Sieves are allowed to use integer variables (system-level Sieves had already been allowed to do so), the following Sieve would work even at user-level to add a manifest to the first text part of the message:

```
require ["variables","mime","foreverypart"];
partnumber=0;
total=0;
addprefix "Manifest:";
foreverypart {
  if not anyof (header :mime :type :is "Content-type" "multipart",
               header :mime :type :is "Content-type" "message")
  {
    partnumber += 1;
    total = total + size;
    if header :mime :contenttype :matches "Content-type" "*" {
      addprefix "Part # " . partnumber . " of type " . "${0}" .
        " and size " . size . " characters";
    }
  }
};
addprefix "Total size: " . size . " characters including header, with";
addprefix "          " . total . " characters of content.";
addprefix "End of manifest.\r\n\r\n";
```

5.1.2.16 Sieve notify extension

New in 7.0.5, the MTA supports [RFC 5435 \(Sieve Email Filtering: Extension for Notifications\)](#) and [RFC 5436 \(Sieve Notification Mechanism: mailto\)](#). The capability identifier for the notify extension defined in [RFC 5435](#) is "enotify". This support extends, rather than replaces, the MTA's existing support (since JES MS 6.2) for the "notify" action defined in [draft-martin-sieve-notify-01.txt](#). Both forms of "notify" can be used simultaneously; e.g.:

```
require ["enotify","notify"];
# New, standard form -- the notify capability
notify :message "subject-text" "mailto:a@b?body=body-text";
# Equivalent older form from the Martin draft -- the notify capability
notify :method "email" :options "a@b" "subject-text" "body-text";
```

If both "notify" extensions are enabled, the action arguments are examined to determine which extension is being used.

New in 7.0.5, "notify" [actions in user-level Sieves are automatically cancelled when the overall Sieve verdict](#) is "jettison", "refuse", "reject", or "ereject". "notify" in a system-level Sieve was treated the same, but no longer: such "notify" actions will now be honored, making it possible to use "notify" for some limited administrative auditing functions.

Note that by default, the "notify" action (both types) is disabled; to enable use of it, the [max_notifys](#) MTA option must be set to a positive value. Also note that as of 7.0.5, the MTA tracks uses of "notify" and limits successive (repeated) such actions within some time period; configuration of this is controlled by various [autoresponse periodicity MTA options](#).

The `enotify` extension also adds two new tests, `valid_notify_method` and `notify_method_capability`. `valid_notify_method` takes an argument consisting of a list of notification methods and returns true if they are supported *and* syntactically valid, or false otherwise. `notify_method_capability` takes three arguments, a notification-uri, a notification-capability, and a list of keywords, and succeeds if a match occurs; this is intended to permit checking whether a desired form of notification can be performed as desired.

The `enotify` extension also adds a modifier `:encodeurl` to the `set` action of the [variables extension](#). `:encodeurl` causes percent-encoding of any octet in the string that doesn't belong to the "unreserved" set for URIs, as described in [RFC 3986 \(Uniform Resource Identifier \(URI\): Generic Syntax\)](#).

Normally, specifying a syntactically invalid recipient address, or syntactically invalid `:from` address, in a "notify" action results in the Sieve script aborting with an error. New in Messaging Server 7.0.5, the [notify_ignore_errors](#) MTA option may be enabled to cause such syntactic errors to be silently ignored.

New in 8.0, the MTA supports a private extension for system-level Sieves "nonotify", which suppresses all applications of either form of the "notify" action. (Technical note: "nonotify" affects those Sieves which are *both*: attached to the same recipient address, *and* evaluated later.)

5.1.2.17 Sieve redirect action

The Sieve "redirect" action performs a type of forwarding of a message. It cannot be combined with any of "refuse", "reject", or "jettison", whose semantics all imply no retention of the message (including no forwarding). The [max_redirects](#) MTA option imposes a limit on the maximum number of "redirect" actions that a user Sieve script is allowed to apply. As of 8.0, this limit applies only to user-level Sieves. Note that when the

MTA performs a "redirect" action, it generates the new message (the redirected, that is, forwarded message) via enqueue to (prior to 8.0) the [reprocess channel](#) or (as of 8.0) the [process channel](#).

A number of extensions can modify the effect of the standard Sieve "redirect" action.

The ["copy" extension](#) defined in [RFC 3894](#) adds the `:copy` parameter to permit "redirect" to take effect without cancelling the default action of saving the message to the "INBOX".

Added in JES MS 6.3p1, the `:resent` and `:noresent` arguments are supported on the [Sieve "redirect" action](#), for controlling whether Sieve "redirect" actions cause addition of Resent-* header lines. The [sieve_redirect_add_resent](#) MTA option may be used to control the MTA's default behavior. See also the [defer_header_addition](#) MTA option, which controls whether Sieve filters see added headers on redirected messages.

New in JES MS 6.3p1, the Sieve "redirect" action supports the `:resetmailfrom` and `:keepmailfrom` parameters, to control whether the envelope FROM for the redirected message is reset to match the Sieve owner, *vs.* the "original" envelope FROM address being retained for use on the redirected message. Note that by DSN rules, `:keepmailfrom` cannot be used when `:notify` or `:ret` are also specified on a "redirect" action.

As of Messaging Server 7.0u2, the `redirect-dsn` extension (capability name `redirect-dsn`) defined in [RFC 6009](#) allows control over delivery status notification (DSN) parameters, adding two new parameters `:notify` and `:ret`. (The `:notify` parameter support was actually added for JES MS 6.3p1, prior to standardization; standardization in [RFC 6009](#) occurred in time for Messaging Server 7.0u2.)

The [Sieve "extlists" extension](#) may be used in conjunction with a "redirect" action, to redirect a message to (an externally stored) list of recipients. The [max_redirect_addresses](#) MTA option imposes a limit on how many such externally stored recipients will be used from the external list.

Note that when the [Sieve environment extension](#) is used, the `vnd.oracle.last-verdict` item is available, and one of its possible values is "redirect" -- which will be the case if and when the [prior Sieve script that applied](#) performed an explicit handling action of "redirect".

5.1.2.18 Sieve relational extension

[RFC 5231 \(Sieve Email Filtering: Relational Extension\)](#) adds relational operators to Sieve conditional tests such as "address", "envelope", and "header". The capability identifier is "relational":

```
require "relational";
```

Relational adds the `:count` match-type permitting counting the number of entities, and the `:value` match-type permitting numeric comparisons of the following forms:

```
:value "gt"
:value "ge"
:value "lt"
:value "le"
:value "eq"
:value "ne"
```

5.1.2.19 Sieve spamtest and virustest extensions

[RFC 5235](#) defines Sieve filter extension tests "spamtest" and "virustest" intended to allow users to test whether a message is spam (unsolicited bulk e-mail) or contains a virus, with the Sieve test syntax itself being independent of the exact mechanism by which the message was determined to be spam or contain a virus. Typically the actual spam/virus determination might be made by a third-party [spam/virus filter package](#) returning a verdict; to convert the underlying spam/virus filter package verdict to a form usable (testable) with "spamtest" and "virustest", the MTA provides private extensions "spamadjust" and "virusset". (The MTA also (7.0u2) supports setting a spam score via the \$, flag in a [FROM_ACCESS mapping table](#) or [recipient * _ACCESS mapping table](#), for cases where, say, a particular message source can be presumed to be emitting spam and/or virii.)

The capability identifiers for the tests from [RFC 5235](#) are "spamtest" (or "spamtestplus" if the ":percent" argument to "spamtest" will be used) and "virustest". The MTA's private "spamadjust" and "virusset" actions are available without any special capability declaration; no "require" action is needed for their use.

Because the intended purpose of "spamtest" and "virustest" is to increase the portability and logical clarity of spam and virus handling in Sieve scripts, insulating the Sieve script from the details of the actual spam or virus detection/determination, understanding the [interaction of multiple Sieve scripts](#) may be of special relevance when setting up to enable use of such tests. For instance, a typical sort of use might be that when a spam/virus filter package returns a string verdict including some spam score, then the MTA is configured to convert that string into a spam score via "spamadjust" action via a corresponding pair of [spamfilterN_verdict_M](#) and [spamfilterN_action_M](#) pair:

```
msconfig> exec get_path "config"
> "/opt/SUNWmsgsr/config"

msconfig> show spamfilter1_*
role.mta.spamfilter1_config_file = /opt/SUNWmsgsr/config/spamassassin.dat
role.mta.spamfilter1_library = IMTA_LIB:libspamass.so
role.mta.spamfilter1_name = SpamAssassin
msconfig> set spamfilter1_verdict_0 False*
msconfig# set spamfilter1_action_0 'require "addheader";virusset "0";addheader "Spam-test: $U";spamadjust "$U";'
msconfig# show spamfilter1_*_0
role.mta.spamfilter1_action_0 = require "addheader";virusset "0";addheader "Spam-test: $U";spamadjust "$U";
role.mta.spamfilter1_verdict_0 = False*
```

Then a user Sieve filter has a spam score available to test. And for instance, one user might choose to configure:

```
require ["fileinto", "spamtest", "virustest", "relational"];
if virustest :value "ge" "3" { discard; }
if spamtest :value "ge" "100"
  { if spamtest :value "ge" "200" { discard; }
    else { fileinto "spam"; }
  }
```

The MTA's support for [Sieve external lists](#) (7.0u1) includes supporting their use (supporting a ":list" argument) in "spamtest" and "virustest" tests. (For an entirely different in details and intention use of "spamadjust" and "spamtest" in conjunction with a Sieve external list, see the example of "white-listing" Personal AddressBook addresses via a [Sieve external list](#); in that example, the external list is a list of addresses, not a list of spam or virus levels.)

As of Messaging Server 7.0u3, the "spamtest" and "virustest" levels in effect for the active Sieve filter for a given recipient will be included in "E" (Enqueue) [MTA message transaction log](#) file entries when the [log_filter](#) MTA option is enabled. This will appear between the Sieve name and the applied action list, *e.g.*:

```
file:///file-spec, spamtest 26.000000, discard
```

5.1.2.20 Sieve subaddress extension

The MTA supports the Sieve subaddress extension specified in [RFC 3598 \(Sieve Email Filtering -- Subaddress Extension\)](#). The capability string is "subaddress":

```
require "subaddress";
```

The subaddress extension adds support for keywords ":user" (the local-part minus the subaddress) and ":detail" (the subaddress itself) to the [address test](#) and, if the [envelope extension](#) has also been enabled, to envelope tests.

Note that configuration of what character the MTA interprets as the separator between the username and their subaddress is controlled by the [subaddress_char](#) MTA option (by default, the plus character, +); background on other aspects of the MTA's subaddress handling can be found in the discussion of [subaddressexact](#) and [related channel options](#).

Use of subaddresses on an email address, whether when [subscribing to mailing lists](#), or for [list moderator purposes](#), or for special purpose message forwarding, can make specialized handling of particular sorts of messages much more convenient. In reaction to the presence of a subaddress, a user's Sieve script might: deliver the message directly into a folder (use "fileinto"), generate an alert notification (use "notify"), perform particular forwarding (use "redirect"), *etc.* For instance, suppose a user has subscribed to a mailing list using the subaddress game-list. Then the following Sieve script:

```
require ["envelope","subaddress","fileinto"];
if envelope :detail "to" "game-list" { fileinto "games"; }
```

would cause messages addressed to the user due to their membership of that list to get filed into the folder named "games".

5.1.2.21 Sieve vacation extension

The MTA supports the standard Sieve extensions "vacation" and "vacation-seconds", defined in [RFC 5230](#) and [RFC 6131](#), respectively, in user-level Sieve filters. The respective capability names are "vacation" and "vacation-seconds", with "vacation-seconds" implying "vacation" which then need not be separately listed in a require clause; it suffices to list:

```
require "vacation-seconds";
```

In addition to the basic ":days" argument for "vacation", and the ":seconds" argument added by "vacation-seconds", the MTA also supports a private argument ":hours" (with the obvious meaning).

The `max_vacations` MTA option specifies the maximum number of Sieve "vacation" actions that may be performed by a Sieve script, with the default being 2. Exceeding this allowed number of vacation actions will result in an error "Too many vacations specified" during Sieve filter evaluation. (Though as of Messaging Server 7.0.5, if `max_vacations=0` is set, then the "require" clause will fail and instead the error upon attempting to use "vacation" or "vacation :seconds" will be "Vacation not listed in require clause prior to use" or "Vacation-seconds not listed in require clause prior to use".)

The MTA supports a private action for system-level Sieves, "novacation", to disable user-level use of "vacation". Use of "vacation" may also be disabled via the [FROM_ACCESS mapping table's \\$! flag](#); indeed, initial configuration of the MTA normally generates [such a FROM_ACCESS mapping table to disable vacation message generation back to typical list "owner" addresses](#). (Technical note: New in 8.0, the effect of "novacation" has been refined a bit. In previous versions, "novacation" only took effect in Sieves evaluated after the one where "novacation" was invoked, and then "novacation" affected *all* subsequent use of "vacation". Since in practice "novacation" is used in the system Sieve to suppress any user Sieve use of vacation, and since the system Sieve is evaluated before any user Sieves, this aspect of "novacation" application had no significant effect. However, as of 8.0, "novacation" now affects those Sieves which are *both*: attached to the same recipient address, *and* evaluated later. This difference in effect is unlikely to matter for "novacation" -- but has been implemented for consistency with "nonotify".)

The MTA supports on "vacation" the private arguments `:reply`, `:echo`, and `:headers`, controlling the format of the response message that the MTA generates. The default, if no such argument is specified, is to generate a [Message Disposition Notification \(MDN\)](#) as specified by [RFC 5230](#). However, `:echo` will produce a "processed" message disposition notification (MDN) that contains the original message as returned content; or `:reply` will produce a pure reply containing only the reply text.

The MTA supports a private argument `:noaddresses` that suppresses the MTA's normal requirement (per the [RFC 5230](#), Section 4.5 requirement) that the recipient address or one of its aliases (an [mailAlternateAddress](#) or [mailEquivalentAddress](#) value, or any alias address specified via the `:addresses` argument) *must* appear in a recipient header line in order for a vacation response message to be generated.

The MTA supports provisioning of users (and domains) with various LDAP attributes that the MTA will interpret as requesting vacation handling: the MTA will convert the values of such LDAP attributes into a pseudo-Sieve script (that is, a "vacation" action), that will be evaluated and applied *before* any explicit Sieve script of the user's. Such LDAP attributes typically have names of the form `mailAutoReply*` or `vacation*` -- for exact names in use, see the MTA options [ldap_start_date](#), [ldap_end_date](#), [ldap_autoreply_mode](#), [ldap_autoreply_subject](#), [ldap_autoreply_text](#), [ldap_autoreply_text_internal](#), [ldap_autoreply_addresses](#), [ldap_autoreply_timeout](#), and [ldap_domain_attr_autoreply_timeout](#).

Previous response tracking, and suppression of additional vacation responses within some time period, is an essential part of "vacation" processing. For additional details on the MTA's implementation of such tracking, see the discussion of the [Autoresponse periodicity MTA options](#).

As of JES MS 6.3, response message generation due to a "vacation" action will be included in [MTA's message transaction log file](#) Sieve filter field when the `log_filter` MTA option is enabled.

As of the 8.0 release, warnings that occur during Sieve evaluation such as issues with the vacation extension (or issues involving the memcache protocol or the [duplicate](#) extension), plus any specifically specified warning text specified via the Sieve ["warn"](#) action will result in a "warn" clause in the [log_filter](#) field of [MTA message transaction log entries](#).

5.1.2.21.1 Why a vacation message was not generated

The list of things that can go "wrong" with vacation messages is pretty much anything that can go wrong with a message in general, plus a lot more vacation-specific factors. For instance:

- the [recipient domain isn't defined properly in LDAP](#), with the result that the [recipient domain isn't found and matched](#) as desired,
- the [recipient address isn't defined properly in LDAP](#), with the result that (even once the recipient domain is found) the [recipient address isn't found and matched](#) as desired,
- the recipient domain or user *is* properly defined and found in LDAP, but currently has a domain or recipient status other than "active" (or a few active-with-special-handling variants such as "defer", "defer-submit", or "deliver"), that is, has a status such as "inactive" or "disabled" or "deleted" or "hold"; see the various domain and user or group status LDAP attributes including [mailDomainStatus](#), [mailUserStatus](#), and [inetMailGroupStatus](#), (or more precisely, the LDAP attributes named, respectively, by the [ldap_domain_attr_mail_status](#), [ldap_user_mail_status](#), and [ldap_group_mail_status](#) MTA options),
- the sender didn't use an expected form of the recipient's address; recall that vacation messages very specifically, (in accordance with what's called for by the standards), don't get sent unless either: the recipient address is "found" in a recipient header line, or another alternate-but-expected address form is "found" in a recipient header line (with such alternate address matching being controlled by the [vacation action's :addresses argument](#) or the LDAP attribute named by the [ldap_autoreply_addresses](#) MTA option),
- the actual original message contains any of various vacation disabling (again, as called for by standards) header lines or notification envelope flags,
- the original message matches a [FROM_ACCESS mapping table entry that sets the \\$! flag](#) or has a system/channel Sieve script apply a ["novacation" action](#),
- the original message came in when the user was not "on vacation" (as specified via the [vacationStartDate](#) and [vacationEndDate](#) LDAP attributes, or more precisely whatever attributes are named by the [ldap_start_date](#) and [ldap_end_date](#) MTA options),
- within the relevant response timeout period, as called for by [RFC 5230 \(Sieve Vacation Extension\)](#) Section 4.2, the "same" vacation message was already sent to this sender; the length of the timeout period is controlled by the [vacation action's :days parameter](#), or via LDAP attributes such as [mailAutoReplyTimeout](#) (or whatever attribute is named by the [ldap_autoreply_timeout](#) MTA option) as modified by the [vacation_minimum_timeout](#) MTA option,
- the MTA encountered problems accessing the [vacation-response-database file](#), or (new in 8.0) problems communicating with [Memcache](#) if the vacation response data is being stored in Memcache,
- a Sieve script attempts to execute "too many" vacation actions, where "too many" is controlled by the [max_vacations](#) MTA option (default 2),

- attempting to use `vacation` from a system level Sieve script,
- attempting to use `vacation` combined with `reject`, `refuse`, or `jettison`,
- [syntax errors in a Sieve vacation action](#),
- for vacation messages defined via LDAP attributes, suitable vacation response text is lacking: the user lacks a value for the `mailAutoreplyText` LDAP attribute, or in the case of an original message sender in the same domain as the user, lacks values for both the `mailAutoreplyText` LDAP attribute and the (preferentially used to respond to other "internal" users) `mailAutoreplyTextInternal` LDAP attribute, (or more precisely, lacks values for the attributes named by the [ldap_autoreply_text](#) and [ldap_autoreply_text_internal](#) MTA options),
- *etc.*

Overall, keep in mind that a vacation message is only supposed to be sent if everything, *absolutely everything*, about a particular message lines up just right as far as that particular message getting a vacation message generated back in response. The fallback in case of problems or doubt about whether a particular message meets all the criteria for sending back a vacation message is to *not* generate a vacation message. See [RFC 3834 \(Recommendations for Automatic Responses to Electronic Mail\)](#) for some of the principles that apply.

As of the 8.0 release, warnings that occur during Sieve evaluation such as issues accessing the vacation-prior-response database (whether that "database" is stored as an on-disk file, or stored in memcache), (as well as similar issues with the [duplicate](#) extension), plus any specifically specified warning text specified via the Sieve "warn" action will result in a "warn" clause in the [log_filter](#) field of [MTA message transaction log entries](#). But most of the above reasons why a vacation message was not generated will *not* result in any such warning, as they are considered part of normal operation.

5.1.2.22 Sieve variables extension

As of JES MS 6.2, the MTA added initial support for the Sieve variables extension, modified in JES MS 6.3 as the initial variables draft changed, eventually to become [RFC 5229](#) (Sieve Email Filtering: Variables Extension). Note that as part of allowing use of string variables, the variables extension also adds to the Sieve language a "set" action and a "string" test. Also, when variables are enabled, the MTA's Sieve implementation allows the use of assignment statements of the forms:

```
var-name := string-expression;  
var-name = string-expression;
```

That is, either "=" or ":=" is supported as the assignment operator. (Note that the semi-colon terminating the statement may be omitted if the statement is at the end of a block.)

The capability string is "variables":

```
require "variables";
```

Several items of note regarding variables:

- Variable substitutions are not allowed in [body test](#) arguments. If they are used, an error is likely to occur. For example, this Sieve script will fail:

```
require ["variables", "body"];
set "a" "testing"
if body :contains "${a}" { discard; }
```

This restriction exists so that a list of all arguments to `body` in all scripts can be computed in advance and searched for in a single pass. If this restriction were to be lifted, it would be easy to construct scripts that would require an arbitrary number of passes over the message, which is unacceptable in a server environment. As such, this should be considered to be a permanent restriction.

- The MTA has a private extension to the [Sieve external lists](#) extension, which is that the MTA also supports properties associated with list entries. When Sieve variables are enabled, properties may be returned in additional variables.
- The "set" action's `:quotewildcard` modifier was first implemented in MS 6.3. However, for MS 7.0.4 the name of the modifier was changed (in the mistaken expectation of a name change in the RFC) to `:quotewild`. As of the 8.0 release, either modifier name, `:quotewild` or `:quotewildcard`, will be accepted.
- When the [Sieve notify extension](#) of [RFC 5435](#) (capability "enotify") is enabled also, it adds an `:encodeurl` modifier to the variables "set" action.
- When the [Sieve regex extension](#) is enabled also, it adds a `:quoteregex` modifier to the variables "set" action.
- [Sieve :regex match type tests](#) now set variables in the same way that `:matches` match type test do. Note that unlike glob-style matches (as when `:matches` is used), where the default is to store whatever matched any wildcard that appears in the pattern, in `:regex` match type tests only those regular expressions enclosed in parentheses are stored. If parentheses are needed but storage is not desired, then the `(?:)` form may be used.
- The maximum number of variables that the MTA will permit in a Sieve script is controlled by the `max_variables` MTA option, by default 128.
- The maximum length that the MTA will permit for a variable name is 128 characters; this is not configurable.
- The MTA does not currently support the use of variable namespaces, so variable names may not contain any periods.
- As of the MS 8.0 release, the MTA supports [user-defined routines \(subroutines\)](#) in [system-level Sieve filters](#). Inside such Sieve subroutines, local variables are supported; local variables are declared in a routine by specifying the "my" control command immediately preceding the first use of the variable; *e.g.*:

```
sub fib(n) {my s = [1, 1];
  my a = 1;
  my b = 1;
  loop {exitif --n < 2;
    my c = a + b;
    s .= c;
    a = b;
    b = c;}
  return s;}
```

- Examples of using variables can be found in discussions of the [Sieve editheader extension](#), [Sieve body extension](#), [Sieve external lists](#), [Sieve mime extension](#), and [Sieve custom tests via mappings](#).

5.1.2.23 Sieve conversiontag extensions

For [system-level Sieve scripts](#), the MTA's private [conversion tag](#) manipulation Sieve actions "addconversiontag" and "setconversiontag" have been supported since JES MS 6.3, and the "removeconversiontag" action has been supported since Messaging Server 7.0u3. No capability name (no "require") clause need be used before using these actions. These actions take a single argument specifying either a string or list of conversion tags. "addconversiontag" adds the specified conversion tag(s) to the current list of conversion tags, while "setconversiontag" empties the existing list before adding the specified new conversion tags. Note that these actions are performed very "late in the game" of message processing, so "setconversiontag" can be used to undo all other conversion tag setting mechanisms. "removeconversiontag" can be used to undo all or part of preceding "addconversiontag" or "setconversiontag" operations.

Also, in [system-level Sieves](#), as of JES MS 6.3, the ["envelope" test accepts "conversiontag" as a field specifier value](#), checking the current list of conversion tags, one at a time. (This test only "sees" the set of conversion tags that were present prior to Sieve processing; the effects of "setconversiontag" and "addconversiontag" are not visible.)

For further discussion and examples of using Sieve conversion tag extensions, see [Sieve filter manipulation of conversion tags](#).

5.1.2.23.1 Sieve filter manipulation of conversion tags

The MTA has a private mechanism of [conversion tags](#), which may be set and used in a variety of ways and for a variety of purposes including special routing, or user-specific automatic document conversion. The MTA's Sieve implementation includes private Sieve extensions to inspect and set these conversion tags. These private Sieve extensions include an [envelope test field conversiontag](#) (new in JES MS 6.3, and supported only in system Sieves) which takes an optional `:count` modifier, as well as several actions supported only in [system Sieve filters](#): the (new in JES MS 6.3) `addconversiontag` and `setconversiontag` actions, and the (new in Messaging Server 7.0u3) `removeconversiontag` action.

The actions `addconversiontag`, `setconversiontag`, and `removeconversiontag` actions take as argument either a Sieve string (containing a single conversion tag value) or Sieve list (consisting of a list of strings each of which is a single conversion tag value). `setconversiontag` clears whatever existing conversion tags might be present, and then adds the specified conversion tag(s). Note that these conversion tag actions are performed relatively late in message processing, so in particular `setconversiontag` can be used to undo all other conversion tag setting mechanisms (including LDAP attribute caused conversion tags).

Note that `removeconversiontag` operates only on conversion tags set via a `setconversiontag` or `addconversiontag` action. However, it is possible by combining operations to have `removeconversiontag` remove a conversion tag present due to some other reason (such as due to a user LDAP attribute having previously set a conversion tag): obtain the current set of conversion tags via the envelope field `conversiontag` and put it into a [Sieve variable](#), then do a `setconversiontag` to the value of that variable, then do a `removeconversiontag`. The reason why this is required (and why

`removeconversiontag` operates the way it does) is to avoid triggering per-recipient message copy split-up: obtaining potentially recipient-specific envelope fields, such as envelope conversion tags, forces per-recipient sensitivity and evaluation of Sieve scripts, and hence potentially forces per-recipient message copy split-up.

For instance, in a [channel Sieve filter](#), if one wanted to remove an "unprocessed" conversion tag (if present) and replace it with a "processed" conversion tag, while preserving any other existing conversion tags, one could use:

```
require ["envelope", "variables"];
if envelope :matches "conversiontag" "*" {
    setconversiontag ${1};
    removeconversiontag "unprocessed";
    addconversiontag "processed";
}
```

Another use of the "envelope" test of the "conversiontag" field is to count how many active conversion tags are present. For instance, if a "high" number of (distinct) conversion tags might as well be considered a request for "full" (do everything) processing, and if there is also a single conversion tag "full-processing" that has that meaning (requesting full processing), then consider:

```
require ["envelope", "relational"];
if envelope :count "ge" "conversiontag" "3" {
    setconversiontag "full-processing"; }
```

5.1.2.24 Sieve `addprefix` and `addsuffix` extensions

The MTA's private `addprefix` and `addsuffix` actions have been supported since Messaging Server 7.0u3. No capability name (no "require") clause need be used before using these actions. These actions take a single string argument specifying text to be added at the beginning or end, respectively, of the first plain text part of a message. (Note that the Sieve language's `text` : syntax for entering long, multi-line strings, may be convenient for specifying prefix or suffix text.)

The effects of "addprefix" or "addsuffix" are similar to the effects of the `mgrpMsgPrefixText` or `mgrpMsgSuffixText` group LDAP attributes (more precisely, the attributes named by the `ldap_prefix_text` and `ldap_suffix_text` MTA options), or the `alias_prefix_text` and `alias_suffix_text` alias options, or the `[PREFIX_TEXT]` and `[SUFFIX_TEXT]` [alias file named parameters](#). Note, however, that the Sieve actions work in any sort of Sieve, not just Sieves attached to groups.

If [multiple Sieves are active](#) and more than one prefix or suffix is specified, they are concatenated.

When logging of Sieve filter actions applied has been enabled via the `log_filter` MTA option, note that only the name of the action, *e.g.*, "addprefix" or "addsuffix", will be included in the logged field; the actual text added to the message will *not* be logged (as it might be very long).

There is an example of use of "addprefix" in the [Sieve editheader extension](#)

5.1.2.25 Sieve `addtag` extension

The MTA's private Sieve extension `addtag` action has been supported since JES MS 6.0. `addtag` provides a convenient way to add a prefix string, that is, a "tag", to a Subject: header line. (The [replaceheader](#) action provides an alternate, though somewhat more complex, mechanism to get such an effect.) Note that the `addtag` effect is *not* visible to other Sieves being evaluated at the same time; this is unlike `addheader` (as of JES MS 6.1p1 and 6.2).

Adding multiple tags is supported, for instance, "Re:" and "FWD:". Prior to JES MS 6.3, `addtag` took a space-delimited list of arguments; as of JES MS 6.3, `addtag` takes a string argument consisting of vertical bar delimited tags. Each of the tags is searched for separately in the current Subject: line, and then added if not already present. (This means in particular that prior to JES MS 6.3, doing an `addtag` operation on a Subject: field that itself includes spaces, e.g., `addtag $U` on a SpamAssassin verdict, was not a good idea: "weird" results were possible when part of the tag string was already present on the Subject: header line. This motivated the change to use of a vertical bar delimiter for JES MS 6.3.)

The `addtag` Sieve action effect is analogous to the effects of the [alias_tag](#) alias option, the [\[TAG\] alias file named parameter](#), or the `mgrpListTag` group LDAP attribute (more precisely, the LDAP attribute named by the `ldap_add_tag` MTA option).

5.1.2.26 Sieve adjustcounter extension

New in 8.0, a set of eight signed, 64 bit counters has been added to the [MTA's counters](#). The values of this set of counters can be adjusted from system Sieve scripts, and later be displayed or accessed via the usual counters display and access facilities. These counters have no predefined meanings; they can be used for any purpose.

A nonstandard Sieve action "adjustcounter" has been added to manipulate these counters, with syntax:

```
adjustcounter [:duplicate] [:channel channel-string] counter [value]
```

where "counter" specifies the counter to operate on (an integer in the range 1-8). "value" is the amount by which to adjust the counter; if omitted, it defaults to 1.

The counters associated with the current source channel are affected by default. The "channel" nonpositional parameter can be used to switch to some other channel. Note that variable substitution can be used on the "channel-string" argument to select a channel computed by the script. Also note that the [channel must be defined in the configuration](#); arbitrary channel names are not allowed.

The "adjustcounter" action can only be used in system-level Sieves; an error will occur if an attempt is made to use it from user-level sieves.

Sieve scripts may be reevaluated multiple times, e.g., when a message is sent to multiple recipients and the script employs an [envelope "to" test](#). When this happens it is normally not desirable for the counter operation to be repeated, so counter adjustments are suppressed by default when scripts are reevaluated. This default can be overridden by specifying the "duplicate" nonpositional parameter.

The counters show up in "imsimta counters -show" output as follows:

```
Sieve counter [1] 1
```

Sieve counter [2]	10
Sieve counter [3]	10
Sieve counter [8]	-15

Note that counters can have negative values. Also note that counters with a value of 0 are suppressed from the display.

These counters can also be retrieved through the `PMDFgetChannelCounters64` routine in the PMDF API.

As an example, the following script fragment, if implemented in a system Sieve on a system that has OpenDKIM set up as a [militer](#), will keep track of DKIM verification operations:

```
require ["variables", "environment"];
if environment :matches "host" "*" {set "host" "${0}";}
if header :matches :index 1 "authentication-results" "*${host}*dkim*" {
  if header :contains :index 1 "authentication-results" "dkim=pass" {
    adjustcounter 1;
  } else {
    adjustcounter 2;
  }
  adjustcounter 3;
} else {
  adjustcounter 4;
}
```

Counter 1 will contain the number of successful verifications performed, counter 2 will contain the number of failed verifications, counter 3 will contain the total verifications, and counter 4 will count the number of messages without a local DKIM result.

5.1.2.27 Sieve capture extension

The MTA supports a private "capture" extension action (capability name "capture" -- but it may be used without explicitly listing it in a "require" clause) for capturing a message copy for legal intercept, or archival, or message replay, or similar purposes. "monitor" is a deprecated synonym for the "capture" action.

Two optional nonpositional parameters, ":dsn" and ":message", were added for JES MS 6.3; ":journal" (to generate Microsoft[®] Exchange "envelope journaling" format) was added for 7.0u2; ":header" (which can be used as a modifier with either :dsn or :journal) was added for 8.0.

See [Format of captured message copies](#) for examples of these formats.

See [Example Sieve external lists with properties](#) for complex examples using "capture".

5.1.2.28 Sieve hold extension

The MTA supports a private "hold" extension action for system Sieve filters; use of this action causes the message file to be side-lined as a `.HELD` file in the MTA queue area. Note that attempts to use the action in a non-system level Sieve script will be ignored without generating an error message.

5.1.2.29 Sieve comparators

In addition to the standard Sieve comparators `"i;ascii-casemap"` and `"i;octet"` described in [RFC 3028](#), and `"i;ascii-numeric"` described in [RFC 4790](#), the MTA also supports some additional, non-standard comparators.

Because the MTA's Sieve implementation supports negative integers in addition to the standard unsigned integers, the MTA supports a `"i;ascii-integer"` comparator to compare signed integers.

The MTA's comparators `"i;ascii-casemap-collapse"` and `"i;octet-collapse"` are similar to the correspondingly named standard comparators, except that all folding white space characters (space, tab, carriage return, line feed) are removed from both the target and pattern strings prior to comparison. Use of these white-space-collapsing comparators is recommended for Sieve comparisons of header values including white space, as many popular email clients exhibit various standards-incompliant behaviors regarding white space in header lines occurring next to MIME encoded-words, or around line folding; standards-compliant Sieve matching may thus not appear to match users' expectations, when client generation and display of white space diverges from standards.

5.1.2.30 Sieve importance extension

New in 7.0.5, the `"importancetest"` Sieve test and `"importanceadjust"` Sieve action have been implemented. These nonstandard extensions are provided so that [multiple Sieve scripts can cooperate](#) in making a determination of a message's importance, much like the `"spamttest"` and `"spamadjust"` extensions allow multiple Sieves to cooperate in determining whether or not a message is spam.

`"importancetest"` and `"importanceadjust"` work in the same way as `"spamttest"` and `"spamadjust"`, except that:

- Importance values range from 0 to 100.
- The initial value of `"importancetest"`, and the value if no `"importanceadjust"` actions have been performed, is 50.
- Fractional adjustments are allowed, but the importance value is rounded to an integer by the `"importancetest"` test.

5.1.2.31 Sieve loop extension

New in 7.0.5, the MTA supports a private `"loop"` construct in system-level Sieves, specifically in spamfilter, in `alias file [FILTER]`, or `alias_filter alias option` defining a non-personal alias (e.g. a mailing list alias), `destination channel`, the `FROM_ACCESS` mapping table or a `Recipient *_ACCESS` mapping table, `source channel`, and system `systemfilter` (`imta.filter`) Sieves. (No `"require"` clause is needed.) The syntax is:

```
loop {  
    ...  
    exitif (expression);  
    ...  
}
```

A loop may contain zero or more `"exitif"` statements. Loops may be nested:

```

loop {
    ...
    loop {
        ...
        exitif (expression1); # Exit from inner loop #1
    }
    ...
    exitif (expression2); # Exit from outer loop
    ...
    loop {
        ...
        exitif (expression3); # Exit from inner loop #2
    }
}

```

Loops should be used with extreme care because of the possibility of putting the MTA into a CPU loop. It tends to be wise to use more limited capabilities, for instance, the ["foreverypart"](#) extension, when such a more limited capability suffices.

5.1.2.32 Sieve memcache extension

As of the 8.0 release, the MTA supports a private "memcache" operator, which may be used to perform both actions and tests. (Configuration of MTA connections to memcache is required; see the [Memcache MTA options](#) and in particular see the [memcache_host](#) MTA option which requires an explicit, valid value.) This nonstandard extension is available without any special capability declaration; no "require" action is needed for its use. By default, the "memcache" operator may be used in any Sieve script, but its availability may be selectively disabled via the [enable_sieve_memcache](#) MTA option. This extension is provided so that Sieve filters can access and manipulate data using the memcache protocol.

The various Sieve "memcache" operations share a set of common nonpositional parameters:

<code>:host host-string</code>	Specifies the host name(s) and port(s) for the memcached server to use in <code>host:port</code> format. The <code>port</code> part is optional and defaults to value given to the memcache_port MTA option. If <code>:host</code> isn't specified, the host and port are given by the memcache_host and memcache_port MTA options respectively.
<code>:host [host-string1, host-string2, ...]</code>	If a list of <code>host:port</code> pairs is specified, one will be chosen by applying a hash function to the key. This provides the means to distribute key-value pairs across multiple memcache servers. Note that the algorithm used is the same as for the memcache mapping callout.
<code>:tableprefix prefix-string</code>	Unlike MeterMaid, each memcache server provides a single storage area; in effect a single "table". Multiple tables can be implemented by adding a prefix to the key. <code>:tableprefix</code> provides a convenient way to specify such a prefix while making it clear that the string is a prefix and not logically part of the key. (Note that some servers that implement the memcache protocol impose additional structure on the key themselves; the <code>:tableprefix</code> parameter may be useful in specifying such structure.)

<code>:timeout timeout-number</code>	Specifies the timeout value in seconds for the entry being created or updated. The default timeout value is 0, which means the entry never times out.
<code>:quota quota-int</code>	Corresponds to MeterMaid throttle table quota setting; see the MeterMaid throttle documentation for additional information.
<code>:quotatimeout quota-timeout-int</code>	Corresponds to MeterMaid throttle table quota timeout setting; see the MeterMaid throttle documentation for additional information.
<code>:penalize penalize-int</code>	Corresponds to MeterMaid throttle table penalize setting; see the MeterMaid throttle documentation for additional information.
<code>:duplicate</code>	It may be necessary to evaluate Sieves more than once, <i>e.g.</i> , when a message is sent to multiple recipients and the Sieve employs an envelope "to" test . Additionally, even if a particular Sieve script does not contain such a test, it may nevertheless be necessary to reevaluate it if a previous script was reevaluated and that action changed one or more of the inputs to the current script.
MATCH-TYPE and COMPARATOR	See RFC 5228 for information about MATCH-TYPE and COMPARATOR parameters.

The memcache Sieve extension provides the following operations:

```
memcache :add [:host host-string] [:tableprefix prefix-string]
           [:duplicate] [:timeout timeout-number] key-string
           value-string
```

Add a new entry with the specified key, value and timeout. The operation only succeeds if the entry does not already exist. The operation returns TRUE if the entry is added successfully, FALSE if not. For example:

```
if not memcache :add "a" "b" { ... handle failure ... }
```

memcache ":add" operations are skipped and return success unconditionally on script reevaluations by default. ":duplicate" forces the operation to be performed on all reevaluations.

```
memcache :adjustdown adjustment-value [:host host-string]
           [:tableprefix prefix-string] [:duplicate] [MATCH-TYPE]
           [COMPARATOR] [:quotatimeout quotatimeout-numeric [:penalize]]
           [:timeout timeout-value] key-string [test-string]
```

Decrement the entry with the specified key by the specified adjustment value. The entry must contain an unsigned decimal string. The adjustment value can be either an integer or a string; if it is a string it must contain an optionally signed decimal value. Negative adjustment value are allowed and will be converted into an appropriate increment operation.

The specified entry will be created (with a value of 0) prior to adjustment if it doesn't already exist. The timeout value is only used if the entry has to be created.

memcache ":adjustdown" operations can be applied to throttle entries. If this is done, the appropriate quota parameters must be specified in case the entry needs to be created.

Note that memcache's increment and decrement operations do not support negative values. Attempts to decrement an entry to a value below 0 will result in a 0 value being stored.

The adjusted value is returned as an unsigned decimal string if neither MATCH-TYPE, COMPARATOR, nor test-string are specified. However, if any of these parameters are specified this operation functions as a test with updated value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to :value "lt" and "i;ascii-numeric" respectively. For example:

```
if memcache :adjustdown 1 "entry" "10" {
    ... handle entry less than 10 ... }
```

memcache ":adjustdown" operations are performed on reevaluations but the adjustment amount is changed to 0. ":duplicate" causes the adjustment amount to be retained on reevaluations.

```
memcache :adjustup adjustment-value [:host host-string]
[:tableprefix prefix-string] [:duplicate] [MATCH-TYPE]
[COMPARATOR] [:quotatimeout quotatimeout-numeric [:penalize]]
[:timeout timeout-value] key-string [test-string]
```

Increment the entry with the specified key by the specified adjustment value. The entry must contain an unsigned decimal string. The adjustment value can be either an integer or a string; if it is a string it must contain an optionally signed decimal value. Negative adjustments values are allowed and will be converted into an appropriate decrement operation.

The specified entry will be created (with a value of 0) prior to adjustment if it doesn't already exist. The timeout value is only used if the entry has to be created.

memcache ":adjustup" operations can be applied to throttle entries. If this is done, the appropriate quota parameters must be specified in case the entry needs to be created.

The adjusted value is returned as an unsigned decimal string if if neither MATCH-TYPE, COMPARATOR, nor test-string are specified. However, if any of these parameters are specified this operation functions as a test with updated value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to :value "gt" and "i;ascii-numeric" respectively. For example:

```
if memcache :adjustup 1 "entry" "10" { ... handle entry greater than 10 ... }
```

memcache ":adjustup" operations are performed on reevaluations but the adjustment amount is changed to 0. ":duplicate" causes the adjustment amount to be retained on reevaluations.

```
memcache :append [:host host-string] [:tableprefix prefix-string]
[:duplicate] key-string value-string
```

Append the specified value to the entry with the specified key. The entry must already exist. Returns TRUE if the operation is successful, FALSE if it is not.

memcache `:append` operations are skipped and return success unconditionally on reevaluations by default. `:duplicate` forces the operation to be performed on all reevaluations.

```
memcache :fetch [:host host-string] [:tableprefix prefix-string]
          [MATCH-TYPE] [COMPARATOR] key-string [test-string]
```

Fetches the value of the entry with the specified key.

The entry's value is simply returned as a string if neither MATCH-TYPE, COMPARATOR, nor test-string are specified. An empty string will be returned if the specified entry doesn't exist.

However, if any of the three parameters are specified this operation functions as a test with current value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to `:is` and `:ascii-casemap` respectively. The test always fails if the entry does not exist. This can be to test for the existence of an entry:

```
if memcache :fetch :matches "entry" "*" { ... entry exists ... }
```

memcache `:fetch` operations are simply repeated on reevaluations.

```
memcache :prepend [:host host-string] [:tableprefix prefix-string]
           [:duplicate] key-string value-string
```

Prepend the specified value to the entry with the specified key. The entry must already exist. Returns TRUE if the operation is successful, FALSE if it is not.

memcache `:prepend` operations are skipped and return success unconditionally on reevaluations by default. `:duplicate` forces the operation to be performed on all reevaluations.

```
memcache :remove [:host host-string] [:tableprefix prefix-string]
            [:duplicate] [:lockout lockout-numeric] key-string
```

Remove the entry with the specified key. A lockout value, if specified, is an unsigned integer specifying the amount of time to "lock" the key - during that time attempts to store an entry with that key will fail. A lockout default of 0 is the default and causes no lockout to occur. Returns TRUE if the operation is successful, FALSE if it is not.

memcache `:remove` operations are skipped and return success unconditionally on reevaluations by default. `:duplicate` forces the operation to be performed on all reevaluations.

```
memcache :replace [:host host-string] [:tableprefix prefix-string]
           [:timeout timeout-value] key-string value-string
```

Update the value and timeout of an entry. The entry must already exist. Returns TRUE if the operation is successful, FALSE if it is not.

memcache `:replace` operations are simply repeated on reevaluations.

```
memcache :store [:host host-string] [:tableprefix prefix-string]
          [:timeout timeout-value] key-string value-string
```

Creates a new entry or updates an existing entry with the specified key, value and timeout. Returns TRUE if the operation is successful, FALSE if it is not.

memcache `:store` operations are simply repeated on reevaluations.

```
memcache :throttle :quota quota-numeric [:duplicate]
          :quotatimeout quotatimeout-numeric [:penalize]
          [:host host-string] [:tableprefix prefix-string]
          [:timeout timeout-value] [MATCH-TYPE] [COMPARATOR]
          [:adjustup adjustment-value] [:adjustdown adjustment-value]
          key-string [test-string]
```

Implements the [MeterMaid](#) throttle capability. See the MeterMaid documentation for details of throttle semantics. Note that since there is no server-side awareness of entry semantics the quota and quotatimeout parameters must be specified in every throttle call in case the entry needs to be created. If the entry already exists the parameter values will be checked against the corresponding values stored in the entry.

The throttle value is incremented by default. Either `:adjustup` or `:adjustdown` can be used to specify an alternate adjustment value. (Note that in this case these nonpositional parameters function as modifiers, not operation specifiers.)

This operation always functions as a test. If neither MATCH-TYPE, COMPARATOR, nor test-string are specified TRUE is returned if the throttle is engaged, FALSE if it is not engaged or an error occurs. If one of these three parameters is provided the throttle is adjusted and then Sieve tests are applied to the value. The default MATCH-TYPE is `:value "gt"` and the default COMPARATOR is `"i;ascii-numeric"`.

memcache `:throttle` operations are performed on reevaluations but the adjustment amount is changed to 0. `:duplicate` causes the adjustment amount to be retained on script reevaluations.

5.1.2.33 Sieve metermaid extension

As of the 8.0 release, the MTA supports a private "metermaid" operator, to access and manipulate data stored in MeterMaid; the operator has uses both as an action and as a test. This nonstandard extension is available without any special capability declaration; no "require" action is needed for its use. By default, the "metermaid" operator may be used in any Sieve script, but its availability may be selectively disabled via the [enable_sieve_metermaid](#) MTA option.

The location of the MeterMaid server that Sieve scripts may query, as well as various other basics of MeterMaid communication, server are specified via [MeterMaid MTA options](#), especially [metermaid_host](#), [metermaid_port](#), and [metermaid_secret](#).

The various Sieve "metermaid" operations share a set of common nonpositional parameters:

```
:host host-string          Specifies the host name and port for the MeterMaid server
                           to use in host:port format. The port part is optional
```

and defaults to value given to the `metermaid_port` MTA option. If `:host` isn't specified, the host and port are given by the `metermaid_host` and `metermaid_port` MTA options respectively.

`:duplicate`

It may be necessary to evaluate Sieves more than once, *e.g.*, when a message is sent to multiple recipients and the Sieve employs an `envelope "to" test`. Additionally, even if a particular Sieve script does not contain such a test, it may nevertheless be necessary to reevaluate it if a previous script was reevaluated and that action changed one or more of the inputs to the current script.

MATCH-TYPE and
COMPARATOR

See [RFC 5228](#) for information about MATCH-TYPE and COMPARATOR parameters.

The following "metermaid" operations are available:

```
metermaid :adjustdown adjustment-value [:host host-string]  
          [:duplicate] [MATCH-TYPE] [COMPARATOR]  
          table-string key-string [test-string]
```

The "metermaid :adjustdown" operation decrements the entry with the given key in the specified table by the adjustment value. This operation is only supported on simple tables configured for integer values and throttle tables.

The adjustment value can be either an integer or a string; if it is a string, it must contain an optionally signed decimal value. Negative adjustment value are allowed and will be converted into an appropriate increment operation.

The specified entry will be created (with a value of 0) prior to adjustment if it doesn't already exist.

The adjusted value is returned as an unsigned decimal string if neither MATCH-TYPE, COMPARATOR, nor `test-string` are specified. However, if any of these parameters are specified, this operation functions as a test with updated value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to `:value "lt"` and `"i;ascii-numeric"` respectively. For example:

```
if metermaid :adjustdown 1 "table" "entry" "10"  
  { ... handle entry less than 10 ... }
```

`:adjustdown` operations are performed on reevaluations but the adjustment amount is changed to 0. `:duplicate` causes the adjustment amount to be retained on reevaluations.

```
metermaid :adjustup adjustment-value [:host host-string]  
          [:duplicate] [MATCH-TYPE] [COMPARATOR]  
          table-string key-string [test-string]
```

The "metermaid :adjustup" operation increments the entry with the specified key in the given table by the adjustment value. This operation is only supported on simple tables configured for integer values and throttle tables.

The adjustment value can be either an integer or a string; if it is a string, it must contain an optionally signed decimal value. Negative adjustments values are allowed and will be converted into an appropriate decrement operation.

The specified entry will be created (with a value of 0) prior to adjustment if it doesn't already exist.

The adjusted value is returned as an unsigned decimal string if neither MATCH-TYPE, COMPARATOR, nor `test-string` are specified. However, if any of these parameters are specified this operation functions as a test with updated value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to `:value "gt"` and `"i;ascii-numeric"` respectively. For example:

```
if metermaid :adjustup 1 "table" "entry" "10"
  { ... handle entry greater than 10 ... }
```

`:adjustup` operations are performed on reevaluations but the adjustment amount is changed to 0. `:duplicate` causes the adjustment amount to be retained on reevaluations.

```
metermaid :fetch [:host host-string] [MATCH-TYPE] [COMPARATOR]
           table-string key-string [test-string]
```

The `"metermaid :fetch"` operation fetches the value of the entry with the specified key from the given table. Fetch is supported on all types of tables, although in the case of throttle tables it is implemented by performing an adjust with an adjustment value of 0.

The entry's value is simply returned as a string if neither MATCH-TYPE, COMPARATOR, nor `test-string` are specified. An empty string will be returned if the specified entry doesn't exist.

However, if any of the three parameters are specified, this operation functions as a test with current value of the entry being the tested value. MATCH-TYPE and COMPARATOR default to `:is` and `"i;ascii-casemap"` respectively. The test always fails if the entry does not exist. This can be used to test for the existence of an entry:

```
if metermaid :fetch :matches "table" "entry" "*" { ... entry exists ... }
```

`:fetch` operations are simply repeated on reevaluations.

```
metermaid :greylisting [:host host-string] [:duplicate]
                [MATCH-TYPE] [COMPARATOR]
                table-string key-string [test-string]
```

The `"metermaid :greylisting"` operation provides access to the [MeterMaid](#) greylisting capability. See the MeterMaid documentation for details of greylisting semantics. Of course this operation only works on greylisting tables.

This operation always functions as a test. If neither MATCH-TYPE, COMPARATOR, nor `test-string` are specified, then TRUE is returned if greylisting is engaged, while FALSE is returned if greylisting is not engaged or an error occurs.

If one of these three parameters is provided, the throttle is incremented and then Sieve tests are applied to the value. The default MATCH-TYPE is `:is` and the default COMPARATOR is

"`i;ascii-casemap`". Note that this is implemented as an adjust operation since the connect operation doesn't provide the throttle value as a result.

The test aspect of "`:greylisting`" operations is repeated on reevaluations, but the entry is not updated. "`:duplicate`" causes the entire operation to be performed on reevaluations.

```
metermaid :remove [:host host-string] [:duplicate]
           table-string key-string
```

The "`metermaid :remove`" operation removes the entry with the specified key from the given table. Returns TRUE if the operation is successful, FALSE if it is not.

Remove is supported on all types of tables.

"`:remove`" operations are skipped and return success unconditionally on reevaluations by default. "`:duplicate`" forces the operation to be performed on all reevaluations.

```
metermaid :store [:host host-string]
              table-string key-string value-string
```

The "`metermaid :store`" operation creates a new entry or updates an existing entry with the specified key in the given table. Returns TRUE if the operation is successful, FALSE if it is not.

Store is supported on all table types except throttle tables.

"`:store`" operations are simply repeated on reevaluations.

```
metermaid :throttle [:host host-string] [:duplicate]
                  [MATCH-TYPE] [COMPARATOR]
                  table-string key-string
```

The "`metermaid :throttle`" operation provides access to the MeterMaid throttle capability. See the [MeterMaid](#) documentation for details of throttle semantics. Of course this operation only works on throttle tables.

This operation always functions as a test. If neither MATCH-TYPE, COMPARATOR, nor `test-string` are specified, then TRUE is returned if the throttle is engaged, while FALSE is returned if the throttle is not engaged or an error occurs.

If one of these three parameters is provided, the throttle is incremented and then Sieve tests are applied to the value. The default MATCH-TYPE is `:value "gt"` and the default COMPARATOR is `"i;ascii-numeric"`. Note that this is implemented as an adjust operation since the connect operation doesn't provide the throttle value as a result.

The test aspect of "`:throttle`" operations is repeated on reevaluations but the entry is not updated. "`:duplicate`" causes the entire operation to be performed on reevaluations.

5.1.2.34 Sieve regex extension

As of JES MS 6.1, the MTA supports the proposed Sieve "regex" extension (see `draft-murchison-sieve-regex-08`), which adds a "`:regex`" match type to the Sieve language. The capability name is "regex":

```
require "regex";
```

Because evaluating arbitrary regex expressions is potentially computationally expensive, whether -- and which -- Sieves may use `:regex` may be controlled with the [enable_sieve_regex](#) MTA option; the default is 1, meaning that `:regex` is supported in all Sieves.

Restrictions on `:regex`: Note that `:regex` is not supported with the `hasflag` test (from the [imap4flags](#) extension). Another restriction is that [utf-8 comparator](#) use with `:regex` is not supported. For performance reasons, the `:regex` match type is not supported for use with the Sieve `body` test.

New in 8.0, Sieve `:regex` match type tests now set [variables](#) in the same way that `:matches` match type tests do. Note that unlike glob-style matches (as when `:matches` is used) where the default is to store whatever matched any wildcard that appears in the pattern, in regex match type tests only those regular expressions enclosed in parentheses are stored. If parentheses are needed but storage is not desired, then the `(?:)` form may be used.

5.1.2.35 Sieve `setenvelopefrom` extension

New in MS 8.0, the MTA supports a private `setenvelopefrom` action in [system level Sieve scripts](#). This nonstandard extension is available without any special capability declaration; no `require` action is needed for its use. `setenvelopefrom` accepts a single string argument specifying the new envelope From address.

For other Sieve actions especially relevant to notification messages, see also the [Sieve `setnotify` and `setreturn` extensions](#).

5.1.2.36 Sieve `setnotify` and `setreturn` extensions

Introduced in Messaging Server 7.0u1, and available in [system Sieves](#) only, are the nonstandard Sieve actions `setnotify` and `setreturn`. `setnotify` specifies a new value for the DSN NOTIFY parameter and `setreturn` specifies a new value for the DSN RET parameter. Both of these parameters are specified in [RFC 3461](#), as are the possible values that can be specified for each one.

The primary use of these actions is expected to be to adjust return policies for suspected spam. For example, if address validation cannot be performed, it may be prudent to disable nondelivery reports, return of content, or both for messages suspected of being spam:

```
setnotify "NEVER";
setreturn "HDRS";
```

For another Sieve action relevant to notification messages, see also the (new in MS 8.0) [`setenvelopefrom`](#) action which, by overriding a message's envelope From address, alters who will be informed if and when a notification message is later generated.

5.1.2.37 Sieve `setoperation` extension

As of the 8.0 release, the MTA supports a private `setoperation` action in system-level Sieves. This nonstandard extension is available without any special capability declaration; no `require` action is needed for its use. This extension is provided so that system-level Sieve

filters can override the type of enqueue operation being performed. Note that this action can be performed on a per-recipient basis.

"setoperation" accepts a single string argument specifying the operation type. The possible operation types are:

```
"relay"  
"submit"  
"passthrough"  
"default"
```

Note that since "setoperation" is a Sieve action, any effects it has only affect MTA behavior *after* Sieve evaluation. In particular, initial header fixups done on receipt of the header occur before Sieve evaluation and are therefore unaffected by this action. Only the [passthrough](#) channel option affects initial header fixups.

Note that for the [Sieve "environment" extension](#), the MTA supports a private item `vnd.oracle.operation-type` to discover a message's existing operation mode.

5.1.2.38 Sieve setpriority and setmtpriority extensions

As of MS 7.0u4, the private "setpriority" action is available for system-level Sieves, to set an [effective message processing priority](#). This nonstandard extension is available without any special capability declaration; no "require" action is needed for its use. "setpriority" takes a single string or integer argument which must be one of "non-urgent", "normal", or "urgent", or an integer in the range 0 to 4; (note that non-urgent corresponds to 2, normal to 3, and urgent to 4). Note that this priority is *not* stored in the message header and only affects processing at this particular stage of message transfer. If multiple "setpriority" actions are specified in different system-level Sieves, the one in the [most specific Sieve](#) wins.

New in MS 8.0, the "setmtpriority" action is available for system-level Sieves. This nonstandard extension is available without any special capability declaration; no "require" action is needed for its use. "setmtpriority" accepts a single integer or string argument and sets the current MT-PRIORITY to the argument value. This action is only allowed in system-level Sieves and the argument must be in the -9 to 9 range of valid MT-PRIORITY values.

5.1.2.39 Sieve translate extension

New in MS 8.0.1, the MTA supports a private "translate" function, to translate a string (or a list) [from one character set to another](#). No "require" clause is needed to use this private extension.

Note that [RFC 5228](#) requires that Sieve filters represent strings in UTF-8. Even strings that (in original messages) were in some other charset are converted to UTF-8 for purposes of Sieve processing. So usually the entire issue of character sets can be ignored for Sieve processing: everything is represented in UTF-8. However, on occasion malformed messages may be encountered that contain incorrectly labelled, or incorrectly structured MIME encoded-words, where proper conversion to UTF-8 cannot be performed. Or special purpose generation of explicitly encoded-in-a-specific-charset strings may be desired. These are cases where the "translate" function may be of use.

"translate" takes three (positional) arguments: the original string, the character set of the original string, and the character set to which to translate the string. Note that the original string remains unchanged; the translated string is returned as a function result.

Either of the following two Sieve sequences of commands will request `fileinto` (folder delivery) to the folder named `noPNOp`. The first example does this making use of a [variable](#), plus the MTA Sieve implementation feature of allowing [an assignment statement](#):

```
require ["variables","fileinto"];
set "string" "abcABC";
rotstring := translate "${string}" "US-ASCII" "ROT13";
fileinto "${rotstring}";
```

The second example does this by making use of the MTA Sieve implementation feature allowing expressions where arguments are normally expected, to perform the `translate` in-line, without use of any named variable:

```
require "fileinto";
fileinto (translate "abcABC" "US-ASCII" "ROT13");
```

5.1.2.40 Sieve warn extension

As of the 8.0 release, warnings that occur during Sieve evaluation (such as memcache protocol issues and issues with the [duplicate](#) or [vacation](#) extensions) will result in a `warn` clause in the `log_filter` field of [MTA message transaction log entries](#).

Additionally, as of 8.0 the `warn` action, which takes either a string or list as an argument, is available for [system-level Sieve scripts](#); its argument will be included in the `warn` clause in the `log_filter` field. This nonstandard extension is available without any special capability declaration; no `require` action is needed for its use.

5.1.2.41 Sieve custom tests via mappings

As of updates to JES MS 6.2 and updates to JES MS 6.3, the MTA's Sieve implementation supports custom Sieve tests defined via [MTA mapping tables](#). (This functionality also existed in an alternate, much less esthetic form, in earlier versions.)

Any mapping table with a name of the form `FILTER_testname` is presumed to define a new, custom Sieve test `testname`. The string result of the mapping will be returned in the `"${0}"` [Sieve variable](#); the flag result of the mapping will be returned in the Sieve `"${1}"` variable.

For instance, a mapping table

```
FILTER_fruitcolor

apple      red-or-yellow-or-green-or-pink$Y
apricot    orange$Y
avocado    green$Y
banana     yellow$Y
blackberry purple$Y
blueberry  blue-or-purple$Y
grape      green-or-red-or-purple$Y
kiwi       green$Y
lemon      yellow$Y
lime       green$Y
```

```
mango      orange$Y
orange     orange$Y
peach      yellow-or-white$Y
pineapple  yellow$Y
raspberry  pink-or-yellow$Y
strawberry red-or-pink$Y
watermelon red-or-pink-or-yellow$Y
*          $N
```

would allow use of a "fruitcolor" test in Sieve; *e.g.*,

```
require ["variables","fileinto"];
if header :matches "Fruit-of-the-day" "*" {set "todaysfruit" "${0}";}
if fruitcolor "${todaysfruit}" {
  set "color" "${0}"; set "flags" "${1}";
  if "${flags}" :is "N" {fileinto "unknown-color-fruits";}
  else {
    if "${color}" :contains "red" {fileinto "red-fruits";}
    if "${color}" :contains "yellow" {fileinto "yellow-fruits";}
    if "${color}" :contains "green" {fileinto "green-fruits";}
  }
}
```

5.1.2.42 Sieve subroutines

(New in MS 8.0.) Support for user-defined routines has been added to the [recipe language](#), and to [system-level Sieves](#). Subroutine definitions have the general form:

```
sub routine-name {routine-body}
```

or if parameters are needed:

```
sub routine-name(parameter1, parameter2, ...) {routine-body}
```

The "return" control command can be used to return a specified result from the routine.

Parameters are passed by value and evaluation of parameters is lazy. If a parameter is never referenced it will never be evaluated. Note that evaluation of parameters in a particular order can be forced very easily:

```
sub f(p1,p2,p3) { p1; p2; p3; ... }
```

Local variables can be declared in a routine by specifying the "my" control command immediately preceding the first use of the [variable](#).

Autoincrement, autodecrement, and the various augmented assignment operators are all allowed on parameters and local variables. So is the exchange operator `:=`. However, exchange cannot be used with a global variable on the right hand side and a local variable or parameter on the left hand side.

For example, a factorial function can be defined as follows:

```
sub f(n) {if n <= 1 {return 1;} else {return f(n-1)*n;}}
```

Recursion is limited to 20 levels. A routine can only call itself recursively since there is currently no forward declaration mechanism.

An example of the use of "my" would be:

```
sub fib(n) {my s = [1, 1];
  my a = 1;
  my b = 1;
  loop {exitif --n < 2;
    my c = a + b;
    s .= c;
    a = b;
    b = c;}
  return s;}
```

Note that use of user-defined routines in Sieves is restricted to system-level Sieves.

5.2 Sieve hierarchy

Messaging Server's Sieve implementation has, as its most significant, major extension to Sieve, a nonstandard one: the ability for multiple scripts to apply to a single recipient. (The Sieve specifications assume a single Sieve script per user.) The MTA supports a number of types of Sieve scripts, whose specified effects are combined to result in an overall Sieve effect for each user.

Sieve scripts come in two general classes, system-level *vs.* user-level. Within these classes there are multiple different [types of Sieve scripts](#). Such multiple Sieve scripts are combined in a logical order, as discussed in [Sieve filters: semantics of multiple scripts](#). Then such multiple Sieve scripts are evaluated in a logical order, as discussed in [Sieve filters: evaluation of multiple scripts](#).

5.2.1 Sieve filters: types of scripts

Messaging Server supports [Sieve scripts of various types specified at multiple levels](#), whose specified effects will then be "combined" to yield an overall Sieve result for each user. The various types of Sieve scripts, in order from the most general to the most specific, are:

1. Spam filter Sieve scripts. Results produced by [spam/virus filter package plugins](#) are interpolated into Sieve scripts. Up to eight such spam/virus filter package plugins can be defined, hence up to eight such Sieve scripts can be produced. Among such spam/virus filter package Sieve scripts, package 1 is the most general proceeding through package 8 as the most specific. (*)
2. Source channel Sieve filters. The [sourcefilter](#) channel option is used to specify the location (via a URL) for a Sieve script applying for messages received via that source channel. (*)

3. System Sieve filter. A single system-wide (per MTA host) Sieve filter can be specified that applies to all recipients of all messages passing through this MTA. In Unified Configuration, the system Sieve filter is specified via the `systemfilter` MTA option; in legacy configuration, the normal location for this script is the file `SERVERROOT/config/imta.filter`. (*)
4. Destination channel script. The `destinationfilter` channel option specifies the location (via a URL) for a Sieve script applying to messages enqueued to this destination channel. (*)
5. `ORIG_SEND_ACCESS`, `SEND_ACCESS`, `ORIG_MAIL_ACCESS`, and `MAIL_ACCESS` mapping table Sieve scripts (in the listed order; that is, the most general being `ORIG_SEND_ACCESS` through the most specific being `MAIL_ACCESS`). The `$$` sequence, when specified in any of these mapping tables, causes a Sieve URL to be read from the mapping result string.
6. Mailing list domain scripts. The domain entry associated with mailing lists defined in LDAP can use the `mailDomainSieveRuleSource` LDAP attribute (more technically, whatever LDAP attribute is named by the `ldap_domain_attr_filter` MTA option) to specify a Sieve script. (*)
7. Mailing list scripts. Mailing lists defined in LDAP can use the `mailSieveRuleSource` LDAP attribute (more technically, whatever LDAP attribute is named by the `ldap_filter` MTA option) to specify a Sieve script. Lists defined via Unified Configuration `alias options` can use the `alias_filter` alias option to specify a Sieve script, just as in legacy configuration lists defined in the `aliases` file or database can use a `[FILTER]` nonpositional parameter to specify a Sieve URL. Also (and note that this is different from the user script case), any "head of household" script applied to a mailing list (specified by having on the group/list LDAP entry the LDAP attributes named by the `ldap_filter_reference` and `ldap_hoh_filter` MTA options) is also considered to be at this same level of generality as other mailing list scripts. The mailing list scripts are considered in the order in which they are encountered.
8. User domain scripts. The domain entry associated with users defined in LDAP can use the `mailDomainSieveRuleSource` attribute (more technically, whatever LDAP attribute is named by the `ldap_domain_attr_filter` MTA option) to specify a Sieve script. (*)
9. User scripts. Users defined in LDAP can use the `mailSieveRuleSource` LDAP attribute (more technically, whatever LDAP attribute is named by the `ldap_filter` MTA option) to specify a Sieve script. In Unified Configuration, users defined via `alias options` can use the `alias_filter` alias option to specify a Sieve script, just as in legacy configuration users defined in the `aliases` file or database can use a `[FILTER]` nonpositional parameter to specify a Sieve URL.
10. **Head of household scripts.** LDAP user entries can contain an attribute (specified by the `ldap_filter_reference` MTA option) that provides the distinguished name of the so-called "head of household", another LDAP user entry. This entry is read and any Sieve stored in the attribute specified by the `ldap_hoh_filter` MTA option (which defaults to `mailSieveRuleSource`) will be processed.
11. Finally, the `filter` channel option can be applied to the destination channel; if used, it specifies a Sieve URL for a user Sieve (that is, it typically specifies a template including user-specific substitutions for how to locate user Sieve scripts).

The types marked with (*) are considered to be "system-level" scripts. Certain capabilities, most notably the "`capture`" action, are only available to system-level scripts. In the other direction, the `vacation action` is only available to user-level (non-system-level) scripts.

Many types of Sieve scripts may be specified or located via a [URL](#); in particular, `file:`, `ldap:`, `data:`, and `imap:` URLs are supported, and a `file:` URL type is normally assumed so bare filenames (the name of a file containing a Sieve script) will also work as an argument in most places.

5.2.2 Sieve filters: semantics of multiple scripts

Since [multiple Sieve scripts](#) can apply to each recipient and different scripts can produce different results, there has to be a way to resolve conflicting results. The rules for determining the final result are:

1. The scripts associated with a particular recipient are scanned in order from most specific to most general. The result of the most specific script that executes an action which determines the status of a message is used preferentially. The actions that determine the status of a message are: 1. `discard` 2. `fileinto` 3. `keep` 4. `redirect` 5. `reject` 6. (Messaging Server 7.0 or later) `ereject`.
2. A set of special, nonstandard actions are provided which, if used, work the other way around: the most general script that specifies them is used preferentially. These special actions are: 1. `jettison` (like `discard`) 2. `refuse` (like `reject`).
3. A Sieve script marked with the (new in 8.0) `override` action becomes the Sieve determining the disposition of a message. If multiple Sieve scripts are marked `override`, then the most general Sieve script "wins". (Thus this is a generalization of the "jettison" and "refuse" sorts of effects -- but in addition to being able to combine "override" with "discard" to obtain a "jettison" effect, or "override" with "reject" to obtain a "refuse" effect, "override" may also be combined with actions such as "fileinto" or "redirect".)
4. "capture" actions in [system-level Sieve scripts](#) are executed unconditionally, regardless of whether or not the Sieve script that contains the "capture" action is selected as the one which determines message handling for this recipient.
5. [Conversion tags](#) set or added by the "`setconversiontag`" and "`addconversiontag`" actions, respectively, are processed unconditionally in a fashion similar to "capture" actions.
6. An error in any Sieve script forces a "keep" action and aborts further scanning. Additionally, a [notification message](#) is sent to the Sieve script owner reporting the problem.

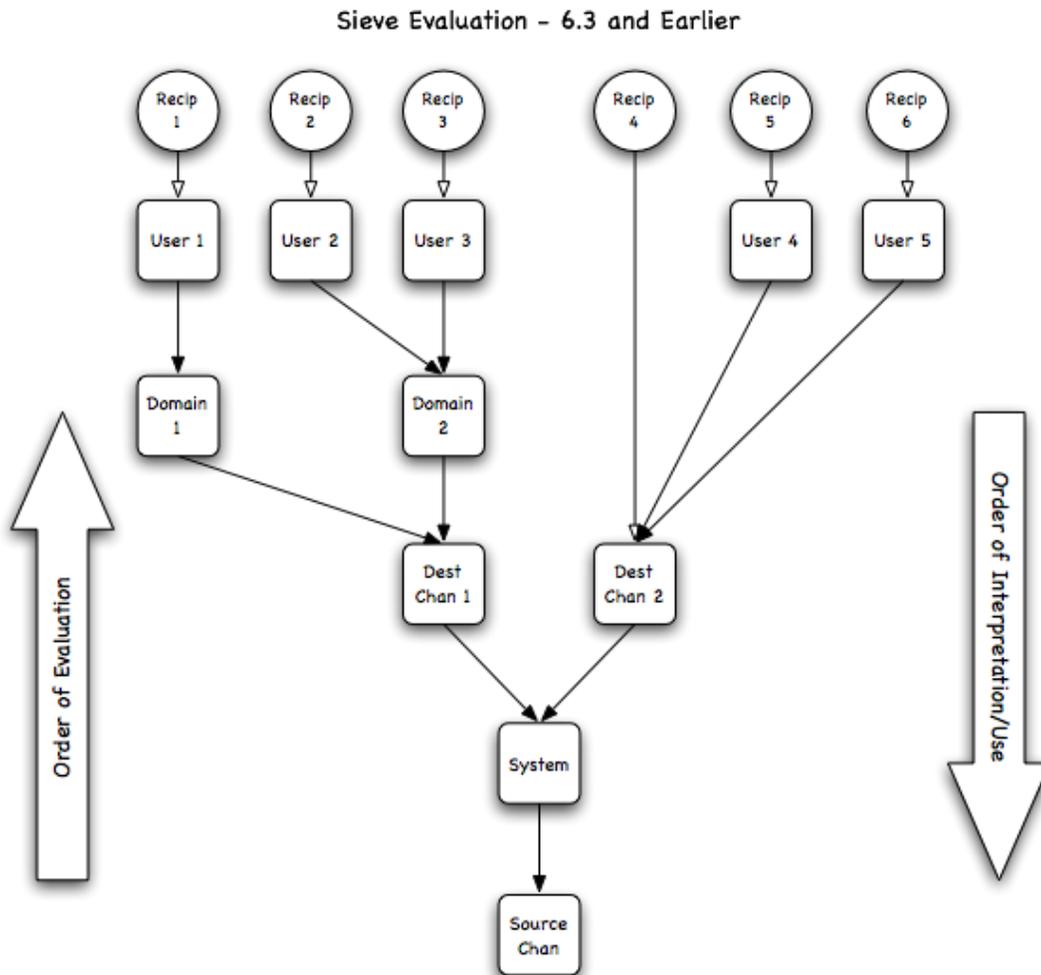
Prior to the 7.0 release, "refuse" actions were only available to system-level Sieve scripts, and "refuse", when used, forced the return of a 5yx response to the DATA command in SMTP. In 7.0 and later this is no longer the case: "refuse" now behaves like "jettison", making it possible for it to apply to only a subset of all recipients. However, the MTA checks and whenever possible will continue to use a 5yz response whenever it is possible to do so.

5.2.3 Sieve filters: evaluation of multiple scripts

The various [different types of Sieve scripts](#) are located, loaded, and associated with the appropriate recipient addresses as early as possible: Source channel scripts and the system Sieve script (`sourcefilter` and `systemfilter`) are dealt with during MAIL FROM processing and all others, with the exception of spam filter Sieves, are dealt with during

RCPT TO processing. Spam filter Sieves (see the `spamfilter*_action*` MTA options) are determined last: since they are derived from spam filter verdict processing, they can only be determined after message data is available.

Prior to Messaging Server 7.0, the internal linkage of Sieve script to recipients looked something like this:



Note that any given tier of the linkage tree can be omitted. As the arrows indicate, evaluation of Sieve scripts proceeded from the bottom to the top, while interpretation of Sieve script results proceeds from the recipients at the leaves down to the root (source channel Sieve filter).

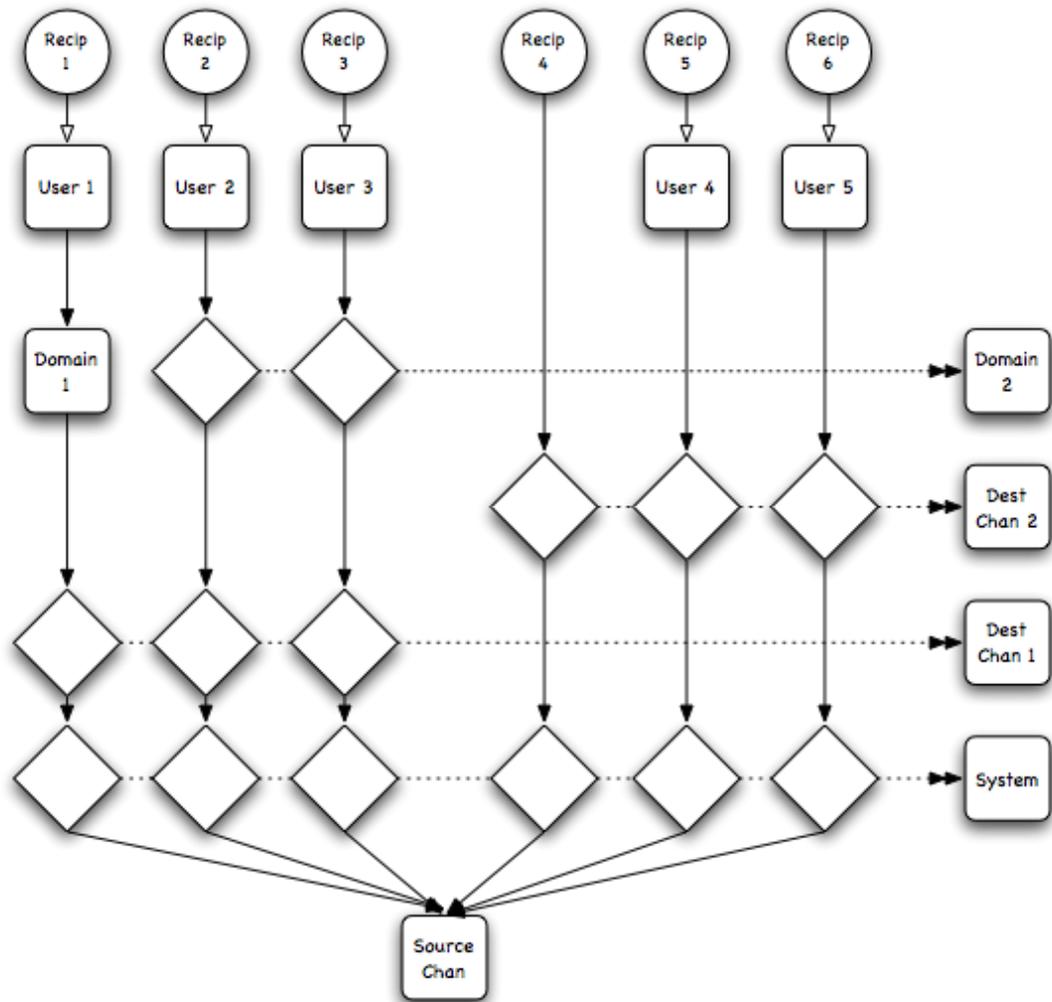
The prior-to-7.0 organization just described was fine in terms of Sieve script evaluation semantics. However, after it was implemented and various additional Sieve extensions were defined, a number of problems emerged:

- Information flow up the tree from the root (source channel Sieve filter) to the leaves (recipient Sieve filters) wasn't possible. This was a nonissue prior to the availability of the `"editheader"`, `"spamttest"`, and `"virustest"` extensions. But once these extensions entered the picture, it was only logical that the effects of more general scripts would be visible to more specific scripts. For example, a system Sieve that performs some test and decides a message is likely to be spam might want to indicate this fact either by adjusting

the spamtest score or by inserting a header. But this doesn't accomplish much when more specific scripts cannot see the results of these actions.

- In some situations, Sieve scripts were loaded but then the recipient addresses they were associated with ended up being dropped from the recipient list. When this happened there was no easy way to remove the scripts from the evaluation list and scripts were evaluated unnecessarily.
- Since the `envelope test` can examine recipient addresses, any Sieve script that employs such a test is necessarily recipient-specific and must be reevaluated for each recipient. Even worse, since this can change the result of the script, it has a cascade effect forcing all more specific scripts to be reevaluated as well. In order to get these semantics, the linkage tree had to be split and dummy nodes had to be inserted. For example, if the system Sieve script for the "envelope" extension, the recipient tree shown above ended up looking like this:

Sieve Evaluation - 6.3 and Earlier
Envelope Test in System Sieve



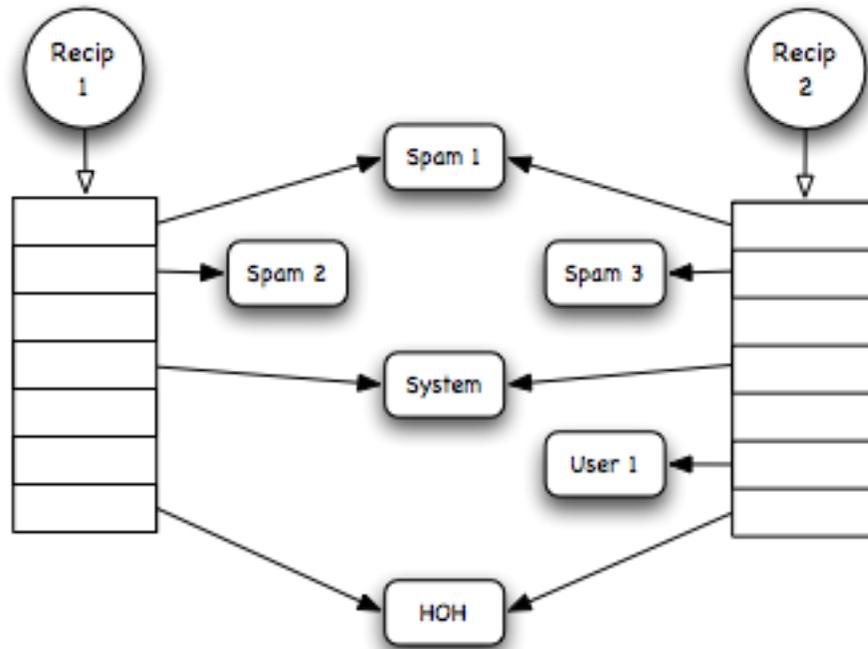
- All Sieve scripts must list all of the extensions they employ in an initial `require` clause. However, it is common for an extension to be listed but not actually used - and this is not necessarily a result of poor coding practice. For example, a script might perform a header or address test and depending on the result of that test only then perform an envelope test, as in the following example where the script is only recipient-specific given certain header values and it is unnecessary to reevaluate it for every recipient. But since the linkage tree has to be constructed prior to script evaluation the presence of the envelope extension in the `require` clause forces unnecessary reevaluations.

```
require ["envelope", "subaddress"];
if address :is "from" "user1@example.com" {
    if envelope :is :detail "to" "whatever" { ... }
}
```

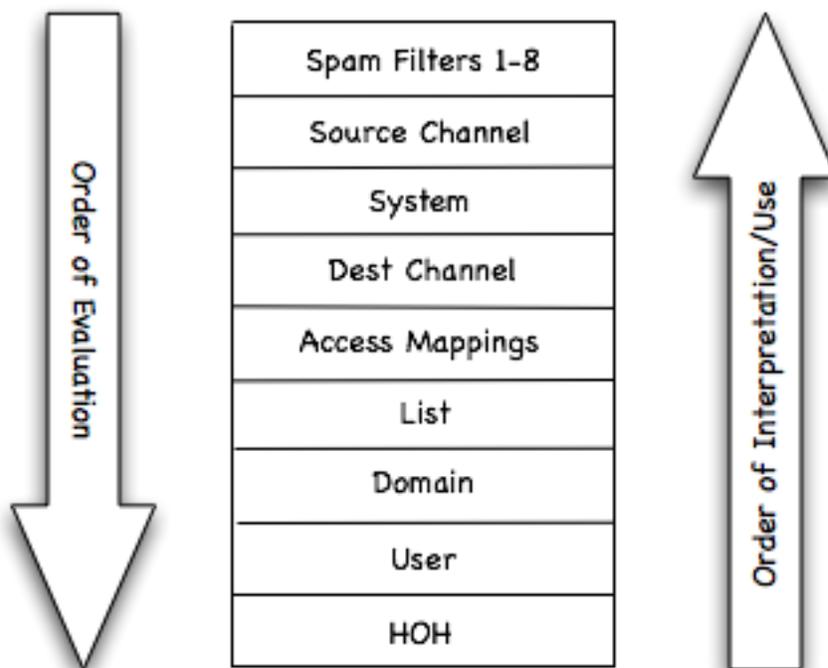
- Sieve scripts often can be written to take advantage of a given extension if it is available but still function if it is not. The approach of enumerating of all extensions in the `require` clause does not allow the construction of such scripts. The `ihave` extension eliminates this restriction by adding a test that succeeds if the requested extension is available and fails if it is not. This would be extremely difficult to implement using the linkage tree approach since the extensions a given Sieve script uses can no longer be determined prior to Sieve script evaluation.

A new way of linking scripts to users was needed and has been implemented in 7.0. The linkage tree is gone, replaced with a per-recipient array:

Sieve Evaluation - 7.0 and Later



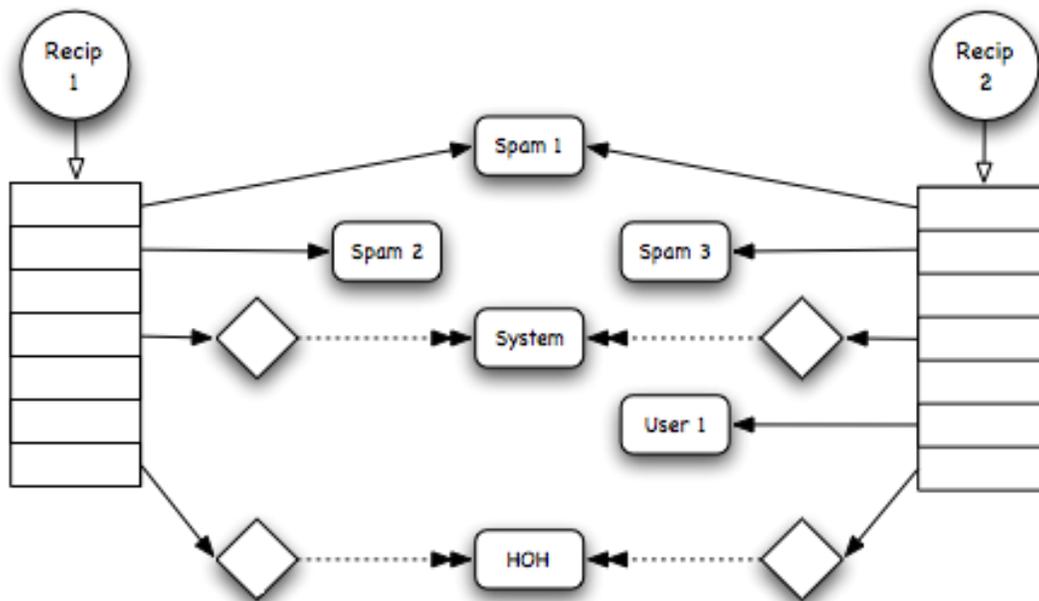
Pointer Array Detail



This new structure eliminates all of the issues the linkage tree had. Scripts are now evaluated during final recipient address processing, eliminating unnecessary evaluation. Evaluation proceeds down the array, and if a particular script has already been evaluated on behalf of some other recipient the results can easily be checked for recipient specificity and reused if no dependency exists. Even better, information can be passed from more general scripts to more specific ones, and the additional checks for recipient-specific information inheritance are reasonably straightforward. And finally, when reevaluation is required the resulting "split" is much more straightforward.

For example, given the previous set of scripts, a recipient-specific system sieve results in the following augmented data structure:

Sieve Evaluation - 7.0 and Later
Envelope Test Evaluated in System Sieve



5.3 Sieve filters: implementation internals

NOTE: This discussion is technical; while it may be of interest during debugging or in discussions with Oracle technical support, it is otherwise unlikely to be relevant or of interest for casual perusal.

Sieve parsing and evaluation is implemented using a generic parse/evaluation subsystem. In addition to Sieve, this system has also been used to implement other languages, most notably the language used by the PMDF-DIRSYNC product. Specific language details, in particular what "functions" can be called and what arguments they require, are specified through callbacks.

Parsed expressions are stored in two separate linked lists of arrays: One for instructions and the other for string data. The use of a series of array segments makes it possible to write parsed

expressions out to disk and read them back in later. This feature is used to store the [system Sieve](#) in the [compiled configuration](#) so it doesn't need to be reparsed.

This subsystem only understands basic script syntax; it knows nothing about specific Sieve semantics. Information about Sieve semantics is provided through callbacks passed to the parser and evaluator. The Sieve-specific callbacks are the routines `mm_check_function` and `mm_eval_function`.

5.4 Head of household Sieve filters

The MTA supports the concept of "head of household" (also referred to as "parental controls") filtering of incoming messages. This refers to cases where the MTA applies a "parent" or "head of household" Sieve filter to a user's incoming messages in addition to the user's own personal [Sieve filter](#). This allows the "parent" or "head of household" user to exert some control over the receipt and handling of messages addressed to the "child" user.

Such "head of household" controls are enabled by marking a "child" user entry with:

- a site-chosen LDAP attribute enabling application of "head of household" control (see the [ldap_parental_controls](#) MTA option), and
- a site-chosen LDAP attribute (see the [ldap_filter_reference](#) MTA option) whose value will be the DN of the entry that contains the actual "head of household" Sieve filter (typically the DN of the "head of household" user).

By default, the "head of household" Sieve filter to be applied will simply be the "head of household" user's own Sieve filter as stored in `mailSieveRuleSource`; note that use of the [Sieve "envelope" extension](#) permits a Sieve filter to be sensitive to the recipient of a message, thus to distinguish between those messages addressed to the "head of household" user him/herself, *vs.* those messages having the "head of household" Sieve filter applied, but which were addressed to some other "child" user. However, see the [ldap_hoh_filter](#) MTA option which may be used to select use of a differently named LDAP attribute as the location of the "head of household" Sieve filter, if it is preferred to store the Sieve filter to be applied in a "head of household" capacity to "child" messages separately from the Sieve filter applying to the "head of household"'s own messages.

Sieve filter application requires a [Sieve "owner"](#) for certain purposes. By default, the "head of household" user's [mail LDAP attribute](#) is taken to be the Sieve "owner" when a Sieve filter is being applied in "head of household" capacity. However, see the [ldap_hoh_owner](#) MTA option which may be used to specify a different LDAP attribute whose value to consider as "owner" for such purposes.

Chapter 6 TCP wrappers

6.1 TCP wrapper filter syntax	6-2
6.1.1 TCP wrapper filter wildcard names	6-4
6.1.2 TCP wrapper filter wildcard patterns	6-4
6.1.3 TCP wrapper filter EXCEPT operator	6-5
6.1.4 TCP wrapper filter server-host specification	6-5
6.2 TCP wrapper filter examples	6-5
6.3 TCP wrapper filter creation	6-7

Access control for clients connecting to Message Store servers (or MMP or proxies) is implemented using the *TCP wrapper* concept. A TCP wrapper 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. The design of the Messaging Server TCP wrapper is based on the Unix Tcpsd access-control facility (created by Wietse Venema).

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)

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, by 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.

See [TCP wrapper filter syntax](#) for a discussion of TCP wrapper filter syntax. Note that [MMP and the proxies](#) use a general `tcpaccess` option to set any combination of Allow and Deny filters, whereas the [IMAP](#) and [POP](#) servers, the [ENS server](#), and the [eval_ldapd server](#), instead use a `domainallowed` option and a `domainnotallowed` option to set, respectively, Allow and Deny filters.

There are also LDAP attributes at the user level, `mailAllowedServiceAccess`, and domain level, `mailDomainAllowedServiceAccess`, that are available to specify per-user or per-domain TCP wrapper access filters. (Note that the MMP and its proxies permit revectoring of exactly what LDAP attribute is used at the user level, via their `tcpaccessattr` option.) See the `ldap_domain_timeout` option for a discussion of the caching of domain level LDAP attributes such as `mailDomainAllowedServiceAccess`.

6.1 TCP wrapper filter syntax

TCP wrapper 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 and host addresses. Both the server and client information can include wildcard names or patterns.

The general syntax of an access filter rule is:

```
< "+" | "-" > service-list: client-list
```

where multiple rules can be placed on the same line, separated by the \$ character, and where

- + (allow filter)¹ means the services in the *service-list* are being granted to the client-list;
- - (deny filter)¹ means the services are being denied to the client list;
- *service-list* is a comma separated list of services to which access is being granted or denied;
- *client-list* is a comma separated list of the clients to be allowed or denied access to the *service-list*.

In more detail, *service-list* is a comma or space separated list of *service-specifications*. A *service-specification* consists of simply a defined *service-name*, or *service-name@host-pattern*, or the special wildcard name ALL, or may make use of combinations of the above with the EXCEPT operator. Defined service names are `imap`, `imaps`, `pop`, `pops`, `smtp`, `smtps`, `http`, `smime`, and as of MS 7.0.5 `mshttpd`.² See [TCP wrapper filter wildcard patterns](#) for more details on *host-patterns*, and [TCP wrapper filter EXCEPT operator](#) for further details on the EXCEPT operator.

client-list is a comma or space separated list of *client-specifications*. A *client-specification* consists of any of a specific *host-name*, a *host-wildcard-pattern*, *username@host-wildcard-pattern*, or in the above forms may make use of the wildcard names described in [Wildcard names for TCP wrapper service filters](#), and optionally the [TCP wrapper filter EXCEPT operator](#).

The very simplest form of a filter is:

```
service: host-specification
```

where *service* is the name of the service (such as `smtp`, `pop`, `imap`, or `http`) and *host-specification* is the host name, IPv4 address, or [wildcard name](#) or [wildcard pattern](#) that represents the client requesting access. When a TCP wrapper filter is processed, if the client seeking access matches *host-specification*, 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 [TCP wrapper filter wildcard names](#).)

Or for a more complex example, making use of a host name in a *service-specification* and a user name in a *client-specification*:

```
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 [TCP wrapper filter server-host specification](#) and [Client User-Name Specification](#).)

When a TCP wrapper filter is processed, if the client seeking access matches *any* of the *client-specification* entries in *client-list*, then access is either allowed or denied (depending on which type of filter this is) to *all* the services specified in *service-list*. 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 [TCP wrapper filter wildcard patterns](#).

The following example enables multiple services on all clients.

```
+imap,pop,http:*
```

The following example shows multiple rules with the `$` rule separator, 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:*
```

The following example shows how to restrict user access so that only SSL-encrypted POP and IMAP are permitted. Because back-end servers do not recognize the `imaps` and `pops` service names,² it is necessary to grant the MMP IP address(es) `pop` and `imap` service access; otherwise, connections between the MMP and the back-end servers will be rejected.

```
+imap,pops:*$+imap,pop:MMP-IP-address(es)
```

¹ Note that use of "+" and "-" in access filters makes sense in values of LDAP attributes such as `mailAllowedServiceAccess` or for components such that MMP which set a general access filter via a `tcpaccess` option; but for components such as the IMAP and POP servers which separately specify Allow and Deny filters (`domainallowed` and `domainnotallowed` options), the explicit "+" and "-" are not needed.

² Note that the MMP supports service names `imap`, `imaps`, `pop`, `pops`, and `smtp`, and `smime`. The back-end (Message Store) system (with its servers such as IMAP and POP) supports `imap`, `pop`, `smtp`, `http`, and `smime`.

6.1.1 TCP wrapper filter wildcard names

[Wildcard names for TCP wrapper service filters](#) shows the TCP wrapper filter wildcard names that may be used to represent service names, host names, or user names.

Table 6.1 Wildcard names for TCP wrapper service filters

Wildcard name	Description
ALL, *	The universal wildcard. Matches all names.
LOCAL	Matches any local short-form host name (one whose name does not contain a dot character). Note that if your deployment uses only canonical names -- fully qualified domain names, hence including dots -- then even local short-form host names will contain dots and thus will not match this wildcard.
UNKNOWN	Matches any host whose name or address is unknown. Use this wildcard name carefully. Host names may be unavailable due to temporary DNS nameserver problems -- in which case all filters that use UNKNOWN will match all client hosts. A network address is unavailable when the software cannot identify the type of network with which it is communicating -- in which case all filters that use UNKNOWN will match all client hosts on that network.
KNOWN	Matches any host whose name <i>and</i> address are known. Use this wildcard name carefully. Host names may be unavailable due to temporary DNS nameserver problems -- in which case all filters that use KNOWN will fail for all client hosts. A network address is unavailable when the software cannot identify the type of network with which it is communicating -- in which case all filters that use KNOWN will fail for all client hosts on that network.
DNSSPOOF	Matches any host whose DNS name does not match its own IP address.

6.1.2 TCP wrapper filter 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 of the form `n.n.n.n/m.m.m.m`. This wildcard pattern is interpreted as a `net/mask` pair. A host IP address is matched if `net` is equal to the bitwise AND of the IP 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`.

- A string of the form `n.n.n.n/p`. This wildcard pattern is interpreted as being in CIDR notation, where *p* is the routing prefix. The corresponding subnet mask, *mask*, is *p* one bits followed by `32-p` zero bits for a total of 32 bits. A host address is matched if the bitwise AND of `n.n.n.n` and *mask* is equal to the bitwise AND of the address and *mask*. For example, the pattern `123.45.67.0/25` matches every address in the range `123.45.67.0` through `123.45.67.127`.

6.1.3 TCP wrapper filter EXCEPT operator

The TCP wrapper 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 *service-list* or *client-list*. 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)
```

6.1.4 TCP wrapper filter server-host specification

You can further identify the specific service being requested in a TCP wrapper filter by including server host name or address information in the *service-specification* entry. In that case the entry has the form *service@host-specification*.

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.

6.2 TCP wrapper filter examples

The examples in this section show a variety of approaches to controlling access using TCP wrapper access filters. 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 TCP wrapper filters, which can improve reliability in the case of nameservice failure.

Example TCP wrapper filter: mostly denying

In this example, access is denied by default. Only explicitly authorized hosts are permitted access.

The default policy (no access) is implemented with a single, trivial deny rule via the `domainnotallowed` option:

```
ALL: ALL
```

This filter denies all service to all clients that have not been explicitly granted access by an Allow filter (set via the `domainallowed` option). The Allow filters, then might be something like these:

```
ALL: LOCAL @netgroup1
ALL: .siroe EXCEPT externalserver.siroe.com
```

The first rule permits access from all short-form host names in the local domain 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`.

Example TCP wrapper filter: mostly allowing

In this example, access is granted by default. Only explicitly specified hosts are denied access.

The default policy (access granted) makes explicit [Allow filters](#) unnecessary. The unwanted clients are listed explicitly in Deny filters (set via the `domainnotallowed` option) 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.

Example TCP wrapper filter: Denying access from spoofed IPs or hosts

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 (which would be set via the `domainnotallowed` option):

```
ALL: DNSSPOOFER
```

This filter denies all services to all remote hosts whose IP addresses don't match their DNS host names.

Example TCP wrapper filter: 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.

Example TCP wrapper filter: Controlling IMAP access while permitting Webmail access

If you wish to allow users to access Webmail, but not access IMAP, create a filter like this:

```
+imap:access-server-host1,access-server-host2
```

This permits IMAP only from the access server hosts *access-server-host1* and *access-server-host2*. You can set the access filter at the IMAP server level by using the option [imap.domainallowed](#), or set the access filter at the user level via the `mailAllowedServiceAccess` LDAP attribute or at the domain level via the `mailDomainAllowedServiceAccess` LDAP attribute. (The MMP and its proxies will at the proxy level use the [tcpaccess](#) option, as well as at the user level whatever user LDAP attribute is named by the [tcpaccessattr](#) option, by default `mailAllowedServiceAccess`, and at the domain level the `mailDomainAllowedServiceAccess` LDAP attribute.)

6.3 TCP wrapper filter creation

The IMAP, POP, and HTTP services support Allow and Deny filters via options [domainallowed](#) and [domainnotallowed](#); (such filters may also be created for SMTP services, but will only apply to authenticated SMTP sessions; instead see [Mail filtering and access control](#) for discussion of the MTA's approaches for controlling access). The MMP and its proxy services, the IMAP Proxy, POP Proxy, SUBMIT Proxy, and vdomain for specifying Virtual Domain specific controls, instead support a [tcpaccess](#) option.

The `msconfig` utility is used to create or edit server level TCP wrapper filters in Unified Configuration; for example:

```
msconfig> set imap.domainallowed .siroe.com
msconfig> set pop.domainnotallowed imap.siroe.com
msconfig> set mmp.tcpaccess "+imap: siroe.com"
```

Note: Restart the relevant services after making changes to their access filters.

TCP wrapper access filters can also be set at the user level or domain level via LDAP attributes; see the user level `mailAllowedServiceAccess` LDAP attribute¹ and the domain level `mailDomainAllowedServiceAccess` LDAP attribute.

¹The MMP and its proxy servers and virtual domains permit renaming of the user level LDAP attribute via their `tcpaccessattr` option. The IMAP, POP, and MSHTTP servers, however, do not support such renaming and expect use of the `mailAllowedServiceAccess` user level LDAP attribute.

Part II Infrastructure

Core infrastructure of Message Server includes the [Scheduler](#) and [Watcher](#), with the Watcher's associated [msprobe](#) facility and [Alarm facility](#), and the [Deployment Map facility](#).

Fundamentals of configuration affecting Messaging Server as a whole, or at least many components, tend to be set as [Base options](#), or for authentication-specific settings, as [Auth options](#).



Chapter 7 Base options

7.1 sslcachedir Option	7-3
7.2 authcachesize Option	7-3
7.3 authcachettl Option Under base	7-3
7.4 bgmax Option	7-4
7.5 bgpenalty Option	7-4
7.6 bgmaxbadness Option	7-4
7.7 bgdecay Option	7-4
7.8 bglinear Option	7-4
7.9 bgexcluded Option	7-4
7.10 debugkeys Option Under base	7-4
7.11 defaultdomain Option Under base	7-4
7.12 stressperiod Option	7-5
7.13 stressfdwait Option	7-5
7.14 ldap_schemalevel option	7-5
7.15 ldap_domain_timeout MTA (and base) option	7-5
7.16 ldap_domain_known_attributes Option	7-5
7.17 ldap_domain_attr_basedn MTA (and base) option	7-6
7.18 ldap_domain_attr_alias MTA (and base) option	7-6
7.19 ldap_domain_attr_uid_separator MTA (and base) option	7-6
7.20 ldap_domain_attr_status MTA (and base) option	7-7
7.21 ldap_domain_attr_mail_status MTA (and base) option	7-7
7.22 ldap_basedn_filter_schema1 and ldap_basedn_filter_schema2 MTA (and base) options	7-7
7.23 ldap_domain_filter_schema* MTA (and base) options	7-8
7.24 accounturl Option	7-8
7.25 filterurl Option	7-8
7.26 folderurl Option	7-8
7.27 installedlanguages Option	7-8
7.28 listurl Option	7-8
7.29 pwchangeurl Option	7-8
7.30 sitelanguage Option	7-9
7.31 dbtxnsync Option	7-9
7.32 enablelastaccess Option	7-9
7.33 hostname Option Under base	7-9
7.34 ldap_host_alias_list Option Under base	7-9
7.35 ldapcheckcert Option	7-9
7.36 ldapconnecttimeout Option	7-10
7.37 ldapmodifytimeout Option	7-10
7.38 ldappoolrefreshinterval Option	7-10
7.39 ldapsearchtimeout Option	7-10
7.40 ldaptrace Option	7-10
7.41 lockdir Option	7-10
7.42 dblockcount Option	7-10
7.43 obsoleteimap Option	7-11
7.44 serveruid Option	7-11
7.45 ssladjustciphersuites Option	7-11
7.46 ssldbpath Option	7-12
7.47 ssldblegacy Option	7-12
7.48 ssldbprefix Option	7-13
7.49 sslcompress Option	7-13

7.50	sslpkix Option	7-13
7.51	sslrequiresafenegotiate Option	7-13
7.52	sslv3enable Option	7-13
7.53	supportedlanguages Option	7-13
7.54	threadolddelay Option	7-13
7.55	tmpdir Option Under base	7-14
7.56	ugldapbasedn Option	7-14
7.57	ugldapbindcred Option	7-14
7.58	ugldapbinddn Option	7-14
7.59	ugldaphost Option	7-14
7.60	ugldapport Option	7-14
7.61	ugldapusessl Option	7-15
7.62	ldaprequiretls Option	7-15
7.63	preferpoll Option	7-15
7.64	secret Option Under base	7-15
7.65	rfc822headerallow8bit Option	7-15
7.66	proxyadmin Option	7-15
7.67	proxyadminpass Option	7-15
7.68	proxyimapport Option	7-16
7.69	proxyimapssl Option	7-16
7.70	proxyserverlist Option	7-16
7.71	proxytrustmailhost Option	7-16
7.72	dcroot Option	7-16
7.73	dnsresolveclient Option	7-16
7.74	loginseparator Option	7-16
7.75	rbac Option	7-16
7.76	listenaddr Option Under base	7-17
7.77	sslnicknames Option Under base	7-17
7.78	welcomemsg Option Under base	7-17
7.79	softtokendir Option	7-17
7.80	ipv6in Option	7-17
7.81	ipv6out Option	7-17
7.82	ipv6usegethostbyname Option	7-18
7.83	ipv6sortorder Option	7-18
7.84	projectid Option Under base	7-18
7.85	tlsv12enable Option	7-18
7.86	sslrenegotiate Option	7-19
7.87	logfile options	7-19
7.87.1	expirytime Option	7-19
7.87.2	flushinterval Option	7-19
7.87.3	filemode Option	7-19
7.87.4	logmillisecond Option	7-19
7.87.5	maxlogfiles Option	7-19
7.87.6	maxlogfilesize Option	7-20
7.87.7	maxlogsize Option	7-20
7.87.8	rollovertime Option	7-20
7.87.9	rolloverpolicy Option	7-20
7.87.10	syslogfacility Option	7-21
7.88	debug Option	7-21
7.88.1	Use with deploymap	7-21
7.88.2	Use with msadmin	7-21
7.88.3	Use with pmxbl	7-21
7.88.4	Use with sms_gateway	7-21

7.88.5 Use with checkpoint	7-22
7.88.6 Use with domainmap	7-22
7.88.7 Use with job controller	7-22
7.88.8 Use with metermaid_client	7-22
7.88.9 Use with dispatcher	7-22
7.89 Base autorestart options	7-22
7.89.1 enable Option Under autorestart	7-23
7.89.2 timeout Option Under autorestart	7-23
7.90 Base certmap options	7-23
7.90.1 dncomps Option	7-23
7.90.2 filtercomps Option	7-23
7.90.3 verifycert Option	7-23
7.90.4 cmapldapattr Option	7-24

Options set at base level tend to be those that either affect overall Messaging Server operation, or else that set a default which may then be overridden for particular services.

Underneath base are also the groups of options `domainmap` (which only has the option `debug`), as well as [autorestart options](#), [certmap options](#).

See also the [umask](#) Message Store option, which affects more than only Message Store files.

7.1 sslcachedir Option

The `sslcachedir` option specifies the SSL session cache directory used to track SSL sessions across multiple connections by the MMP. Prior to 7.0.5.31.0, this also controlled the location of the SSL database files and defaulted to the config directory. As of the 7.0.5.31.0 release, the [ssldbpath](#) base option takes precedence over this option for specifying the location of SSL database files.

NOTE: In order for results to be predictable, this option must be the same for the IMAP, POP, and SMTP proxies -- that is, in legacy configuration it must be the same in the files `ImapProxyAService.cfg`, `PopProxyAService.cfg` and `SmtpproxyAService.cfg`, or in Unified Configuration any settings of this option for the various proxies must match.

7.2 authcachesize Option

The `authcachesize` base option specifies the maximum number of concurrent users/entries in the user/authentication cache.

The Messaging Server can cache the results of LDAP user lookups and successful authentication (*e.g.*, when logging into IMAP, POP or SMTP). The `authcachesize` option defines the number of authentication user cache entries. A higher setting for `authcachesize` improves performance while using more memory. A lower setting reduces performance and reduces the amount of memory used.

7.3 authcachettl Option Under base

The `authcachettl` base option specifies the length of time in seconds an authentication cache entry will remain valid. Set to 0 to disable authentication caching.

Note that setting `ldapcachettl` smaller than `authcachettl` causes the entire user entry to expire, thereby also expiring the user authentication information in the user entry.

7.4 bgmax Option

The `bgmax` option specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See `bgpenalty` for more information.

7.5 bgpenalty Option

When an authentication failure occurs from a particular client IP address, subsequent authentication attempts from that IP address are treated as "BadGuys" and are delayed. If an authentication failure is followed by a successful authentication, the successful authentication is delayed, but the IP address ceases to be treated as a "BadGuy" for subsequent attempts.

`bgpenalty` is the length of time in seconds added to the authentication delay after each failed authentication.

7.6 bgmaxbadness Option

The `bgmaxbadness` option specifies the maximum length of time in seconds for the authentication delay which occurs after a series of failed authentication attempts. See `bgpenalty` for more information.

7.7 bgdecay Option

The `bgdecay` option represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See `bgpenalty` for more information.

7.8 bglinear Option

The `bglinear` option defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See `bgpenalty` for more information.

7.9 bgexcluded Option

The `bgexcluded` option represents a list of excluded IP/mask pairs, or the name of a file to read for these pairs. These client addresses will not be penalized for authentication failure (there is no default value).

7.10 debugkeys Option Under base

The `debugkeys` base option specifies a space-separated list of keywords used to enable various optional debugging facilities.

7.11 defaultdomain Option Under base

The `defaultdomain` base option specifies the Messaging Server default domain. This is used to determine whether a domain is the default domain or a hosted domain.

Normally the `defaultdomain` base option is set to an appropriate value during initial configuration.

The MTA has a "twin" option, `ldap_default_domain`, that can override the `defaultdomain` base option for MTA purposes. See the description of `ldap_default_domain` for details on how the MTA uses the `defaultdomain` value (if `ldap_default_domain` is not set).

7.12 stressperiod Option

When a process such as the MMP becomes stressed due to high load or a denial of service attack, the `stressperiod` option controls how long (in seconds) the Watcher will consider that process stressed as well as how often that process will send a new stress notification to the Watcher. This allows `msprobe` to tell the difference between a wedged server (which should be restarted to improve the system) and a stressed process that is making forward progress but may have a response time larger than the `msprobe` timeout. Restarting such a process may reduce overall system performance as any disconnected clients are likely to reconnect to the system thus increasing the load. If a stressed process becomes wedged, then `msprobe` will be able to restart that process after the `stressperiod` expires.

This is presently only implemented by the MMP.

7.13 stressfdwait Option

When a process is running out of available file descriptors and the `stressfdwait` Base option is set, then the process is permitted to stop accepting new connections until the file descriptor shortage goes away. If this is turned off, the process will continue trying to accept connections until it fails. This is presently only implemented for the MMP and is on by default. As `stressfdwait` is a new feature, it may be helpful to disable the feature temporarily if it is causing an unexpected problem.

7.14 LDAP bind and connect options: `ldap_schemalevel` (1 or 2)

The `ldap_schemalevel` [base option](#) specifies the schema level in use. This option is also available at MTA level. Supported values are 1 or 2. If this option is not set, schema level 1 is assumed to be in use.

7.15 LDAP lookup cache MTA options: `ldap_domain_timeout` (integer)

The `ldap_domain_timeout` option (available at both base and MTA levels) controls the retention time (in seconds) for entries in the domain map cache. The default is -900; as the value used is the absolute value of the `ldap_domain_timeout` setting, this corresponds to 15 minutes. If setting `ldap_domain_timeout` explicitly, set it to a positive value so that the MTA can detect that it has indeed been intentionally set.

7.16 `ldap_domain_known_attributes` Option

The `ldap_domain_known_attributes` MTA (and [base](#)) option controls whether the MTA's domain lookup LDAP queries request all domain attributes, *vs.* solely a hard-coded list of "known" domain attributes. The default of `-1` means to request all domain attributes; setting this option to `1` causes the MTA to request its hard-coded list of "known" domain attributes.

It has been claimed that the `ldap_domain_known_attributes` setting can have a performance impact in some LDAP server environments.

Note that if a site has configured the MTA to use any site-specific custom LDAP attributes, in addition to the normal set that the MTA is hard-coded to interpret, then it is important to use the default setting of `-1` so that LDAP domain queries will return those additional, custom LDAP attribute values.

7.17 Direct LDAP attribute name MTA options: `ldap_domain_attr_basedn` (LDAP attribute name)

The `ldap_domain_attr_basedn` MTA (and [base](#)) option names the domain LDAP attribute, by default `inetDomainBaseDn`, used to store the base DN for the domain's users and groups.

The presence in a domain entry of the attribute named by `ldap_domain_attr_basedn` is not always obligatory with Schema 2, as with Schema 2 in the domain attribute's absence user and group entries will be assumed to reside directly under the domain entry.

Note that the mapping table domain map attribute substitution `}${domain,_base_dn_{` returns either the value of the LDAP attribute named by the `ldap_domain_attr_basedn` MTA option (so normally the value of the `inetDomainBaseDN` LDAP attribute), or if no such LDAP attribute is set as can be the case in Schema 2 mode, returns a constructed DN corresponding to the DN for the domain entry.

7.18 Direct LDAP attribute name MTA options: `ldap_domain_attr_alias` (LDAP attribute name)

The `ldap_domain_attr_alias` MTA (and [base](#)) option specifies the name of an LDAP attribute (by default `aliasedObjectName`) used to identify domain alias entries in the directory. The attribute is present only on a domain alias entry, not on the canonical domain entry; it contains the DN of the entry for which it is an alias. It is used only in Schema 1 or in Schema 2 compatibility mode (with a DC Tree), not in Schema 2 native mode (no DC Tree).

7.19 Direct LDAP attribute name MTA options: `ldap_domain_attr_uid_separator` (LDAP attribute name)

The `ldap_domain_attr_uid_separator` MTA (and [base](#)) option names the domain LDAP attribute, by default `domainUidSeparator`, used to store what the separator character is

between UIDs and domains for addresses in this domain. This option is used both by the MTA, and by the authentication code; the authentication code looks first for the option to be set at base level, but if not set there, the authentication code will use the MTA level option setting.

7.20 Direct LDAP attribute name MTA options: ldap_domain_attr_status (LDAP attribute name)

The `ldap_domain_attr_status` MTA (and `base`) option names the domain LDAP attribute, by default `inetDomainStatus`, whose value specifies the current status of the domain. (The analogous user level attribute is `inetUserStatus` or whatever user LDAP attribute is named by the `ldap_user_status` MTA option; an analogous group attribute can be defined via the `ldap_group_status` MTA option. Compare also with the `ldap_domain_attr_mail_status` MTA option naming the domain LDAP attribute specifying the current mail status of the domain.) Valid values for the attribute named by the `ldap_domain_attr_status` option are `active`, `inactive`, or `deleted`. If no such attribute is present, or is present but with no value, a value of `active` is assumed.

7.21 Direct LDAP attribute name MTA options: ldap_domain_attr_mail_status (LDAP attribute name)

The `ldap_domain_attr_mail_status` MTA (and `base`) option names the domain LDAP attribute, by default `mailDomainStatus`, whose value specifies the current mail status of the domain. (The analogous user level attribute is `mailUserStatus` or whatever user LDAP attribute is named by the `ldap_user_mail_status` MTA option; the analogous group attribute is `inetMailGroupStatus` or whatever group LDAP attribute is named by the `ldap_group_mail_status` MTA option. Compare also with the `ldap_domain_attr_status` MTA option naming the domain LDAP attribute specifying the current general status of the domain.) Valid values for the attribute named by the `ldap_domain_attr_mail_status` option are: `active`, `inactive`, `deleted`, `hold`, `disabled`, `overquota`, and (new in JES MS 6.0) `unused` and `removed` and (new in 8.0) `nonlocal`; other values are interpreted as `inactive`. Note that the `imquotacheck` utility is what updates `mailDomainStatus` to set it to `overquota`.

7.22 Direct LDAP schema MTA options: ldap_basedn_filter_schema1 (LDAP URL filter), ldap_basedn_filter_schema2 (LDAP URL filter elements)

(New in JES MS 6.3-0.15.) The `ldap_basedn_filter_schema1` MTA option specifies the filter used to identify schema 1 domains when performing baseDN searches. The `ldap_basedn_filter_schema2` MTA option specifies additional filter elements used to identify schema 2 domains when performing baseDN searches.

ldap_domain_filter_schema*
MTA (and base) options

The default is that neither the `ldap_basedn_filter_schema1` MTA option nor `ldap_basedn_filter_schema2` MTA option is set. When these options are not set, then the values of `ldap_domain_filter_schema1` and `ldap_domain_filter_schema2`, respectively, are used if those options are set. But if none of these options are set, then the default for `ldap_basedn_filter_schema1` is "(objectclass=inetDomain)", while the default for `ldap_basedn_filter_schema2` is the empty string.

7.23 Direct LDAP schema MTA options:

`ldap_domain_filter_schema1` (LDAP URL filter), `ldap_domain_filter_schema2` (LDAP URL filter)

The default is that neither the `ldap_domain_filter_schema1` nor `ldap_domain_filter_schema2` option is set, neither at the MTA level nor at the [base](#) level. When these options are not set, then internal defaults in the domain map code are used, equivalent to:

```
ldap_domain_filter_schema1=(|(objectclass=inetDomain)(objectclass=inetdomainalias))  
ldap_domain_filter_schema2=(objectclass=sunManagedOrganization)
```

7.24 `accounturl` Option

The `accounturl` [base option](#) specifies the location of the server administration resource for end users (obsolete).

7.25 `filterurl` Option

The `filterurl` [base option](#) specifies the URL for incoming mail (server side) filter (obsolete).

7.26 `folderurl` Option

The `folderurl` [base option](#) specifies the URL for personal folder management (obsolete).

7.27 `installedlanguages` Option

The `installedlanguages` [base option](#) takes a comma separated list of language codes: alphabetic characters only, comma separated list (e.g. "en, fr"). This is identical to [RFC 2068](#)'s Accept-Language: field definition, but with no q-value.

7.28 `listurl` Option

The `listurl` [base option](#) specifies the URL for mailing list management (obsolete).

7.29 `pwchangeurl` Option

The `pwchangeurl` [base option](#) specifies the URL a user visits to change his/her password. If specified, this will be sent with password expiration warnings (e.g., IMAP ALERT).

7.30 sitelanguage Option

The `sitelanguage` base option specifies the default language tag.

7.31 dbtxnsync Option

The `dbtxnsync` [base option](#) sets the database transaction synchronization level. A value of 0 or 1 selects none while a value of 2 requests that all writes be synchronously flushed to the log on every transaction commit.

7.32 enablelastaccess Option

The `enablelastaccess` [base option](#) enables last access time tracking. Access time data is used by `imsconnutil` and `mboxutil -o -t`.

7.33 hostname Option Under base

The `hostname` [base option](#) specifies the fully qualified DNS hostname of this mail server. Normally this option is set to an appropriate value during initial configuration.

The MTA has a "twin" option, `ldap_local_host`, that may be set to override this base option for MTA-specific purposes.

7.34 ldap_host_alias_list Option Under base

The `ldap_host_alias_list` Base option allows setting a list of host names that will be recognized as synonyms for the host name. It corresponds to the `configutil` parameter `local.imta.hostnamealiases` in legacy configuration. The value takes a comma-separated list of up to 40 host aliases; each host alias may be at most 256 characters long; the total length of the entire list is limited to 1024 characters. (In iMS 5.2, the limits were smaller: at most 20 host aliases and each host alias at most 252 characters long.)

In Unified Configuration, this value is used by the mail store when interpreting the `mailHost` attribute to determine whether a user's mailboxes can be accessed locally. In legacy configuration, the `local.imta.hostnamealiases` must be used for this purpose.

Unless `mta.local_host_alias_list` has been set (thereby overriding the `base.local_host_alias_list`), the `local_host_alias_list` base option also affects MTA operation. The `ldap_host_alias_list` value(s) are used by the MTA when deciding whether a domain's `mailRoutingHosts` value(s) or a user's `mailHost` value is "local" (this MTA itself). That is, once an LDAP lookup of a domain or user occurs, this option's value(s) affect the interpretation of the result of the LDAP lookup.

7.35 ldapcheckcert Option

The `ldapcheckcert` base option controls whether to verify the LDAP server certificate.

7.36 ldapconnecttimeout Option

The `ldapconnecttimeout` base option specifies the time in seconds to wait for a new LDAP connection to complete.

7.37 ldapmodifytimeout Option

The `ldapmodifytimeout` base option specifies the time in seconds to wait for LDAP modify operations to complete.

7.38 ldappoolrefreshinterval Option

The `ldappoolrefreshinterval` base option specifies the length of time in minutes before LDAP connections are automatically closed then re-established to the LDAP server. Also, length of elapsed time in minutes until the failover directory server reverts back to the primary directory server. If set to -1, use the code default which is 35 minutes.

7.39 ldapsearchtimeout Option

The `ldapsearchtimeout` base option specifies how many seconds the server will wait for an LDAP search to complete (unless there is a more specific timeout option for that LDAP search), before the search will failover to a backup LDAP server or the operation will fail.

Note that the MTA has its own configuration setting, `ldap_timeout`.

7.40 ldaptrace Option

The `ldaptrace` base option enables LDAP trace (debug) logging. Deprecated in favor of `debugkeys` (Unified Configuration) or `local.debugkeys 'ldap'` key.

7.41 lockdir Option

The `lockdir` base option specifies the full pathname of server lock directory. Defaults to `/tmp/.ENCODED_SERVERROOT/lock/` on Solaris and `/dev/shm/.ENCODED_SERVERROOT/lock/` on Linux, where `ENCODED_SERVERROOT` is composed of mail server user plus the `$SERVERROOT` with `/` replaced by `_`. *e.g.*, `/tmp/.mailsrv_opt_sun_comms_messaging64/lock/`

If the directory does not exist, it will be created.

On Linux, it is a better choice to locate the directory under `/dev/shm` rather than under `/tmp`.

IMPORTANT: Stop and restart all Message Store processes immediately after changing this value. New processes will use the new value and be unable to communicate with a stored using the old value.

7.42 dblockcount Option

The `dblockcount` base option sets the maximum number of BDB locks. The minimum allowed value is 5000; the maximum is 500000.

7.43 obsoleteimap Option

The `obsoleteimap` base option allows use of old IMAP2bis and IMAP4 commands.

7.44 serveruid Option

The `serveruid` base option specifies the UNIX user id of Messaging Server. This is deprecated in favor of the `user` option in `restricted.cnf` which is used preferentially. In Messaging Server 8, this option is always ignored and `configutil -o local.serveruid` returns the value of the `user` option in `restricted.cnf`.

7.45 ssladjustciphersuites Option

The `ssladjustciphersuites` option allows adjusting which SSL cipher suites are enabled or disabled. SSL cipher suites control the level of protection required between SSL client and server. Different cipher suites have different properties and use different cryptographic algorithms. At any time a specific cryptographic algorithm might be weakened or compromised by new research in cryptography. The ability to change the default cipher suites allows the software to adapt as security technology changes. In addition as CPUs get faster, the key size necessary to provide several years of comfortable protection increases, even if the algorithm is considered state-of-the-art.

The default set of SSL cipher suites used may change over time as more secure ones are introduced and weaker ones are deprecated. It is expected most deployments will be happy with the default set of cipher suites and it is generally not a good idea to adjust the available cipher suites without reason. However, here are some scenarios where it may be helpful to adjust cipher suites:

1. a site with specific security policies may wish to provide a fixed list of cipher suites to use that is set by site policy rather than simply using state-of-the-art suites provided by the NSS library. Such a site would typically configure this setting to `'-ALL,...'` where `'...'` contains the cipher suite names.
2. A site which is experimenting with higher performance or more secure cipher suites that require installation of special server certificate types, for example the elliptic curve cipher suites. Such a site would enable these additional suites once installation was complete using a setting such as `'+TLS_ECDH_RSA_WITH_AES_128_CBC_SHA'` to enable an ECDH_RSA cipher suite from [RFC 4492](#).
3. If a site is forced to continue supporting a particularly old client that only supports weak cipher suites, they can be explicitly enabled (for example `'WEAK+DES'` enables the single-DES cipher suites).
4. In the event the cryptographic research community discovers a vulnerability in one or more of the ciphers enabled by default, this provides a mechanism to immediately disable those ciphers. For example, to disable all ciphers using the 'RC4' algorithm, simply set `'-RC4'`.

As of 2008-Jan-29, the available cipher suites in the NSS library are as follows:
`TLS_DHE_RSA_WITH_AES_256_CBC_SHA`, `TLS_DHE_DSS_WITH_AES_256_CBC_SHA`,

TLS_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_RC4_128_SHA,
 TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
 SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
 TLS_RSA_WITH_AES_128_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
 SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA,
 SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA,
 SSL_DHE_DSS_WITH_DES_CBC_SHA, SSL_RSA_FIPS_WITH_DES_CBC_SHA,
 SSL_RSA_WITH_DES_CBC_SHA, TLS_RSA_EXPORT1024_WITH_RC4_56_SHA,
 TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA, SSL_RSA_EXPORT_WITH_RC4_40_MD5,
 SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5, SSL_RSA_WITH_NULL_SHA,
 SSL_RSA_WITH_NULL_MD5, TLS_ECDH_ECDSA_WITH_NULL_SHA,
 TLS_ECDH_ECDSA_WITH_RC4_128_SHA,
 TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
 TLS_ECDHE_ECDSA_WITH_NULL_SHA, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,
 TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDH_RSA_WITH_NULL_SHA,
 TLS_ECDH_RSA_WITH_RC4_128_SHA, TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,
 TLS_ECDHE_RSA_WITH_NULL_SHA, TLS_ECDHE_RSA_WITH_RC4_128_SHA,
 TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA. This list excludes the SSL2 cipher suites as
 Messaging Server has not supported SSL2 since the 6.0 release. While the standard names
 for cipher suites (as published in TLS RFCs) are preferred, there is limited support for legacy
 names used in previous releases and for some OpenSSL names. Note that the TLS_DHE_*
 cipher suites are only available for outgoing connections from Messaging Server.

Starting with version 7.0.5.31.0 (NSS 3.16), the following additional cipher suites are available when the `tlsv12enable` option is set:

TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
 TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,
 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA256,
 TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_NULL_SHA256,
 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.

7.46 ssldbpath Option

The `ssldbpath` base option specifies the location of certificate and key files. This defaults to the product configuration directory.

7.47 ssldblegacy Option

When set, the `ssldblegacy` base option requires SSL/TLS server certificates, CAs and CRLs to be stored in the legacy `cert8.db` and `key3.db` formats supported by our SSL/TLS library. When this is not set, both the legacy `cert8.db/key3.db` and the modern `cert9.db/key4.db` formats are supported. The legacy format requires servers to be shut down when updating

the database for any reason. The modern format allows updates to be performed while the servers are running. The default was changed from 1 to 0 in the 7.0.5 release. Starting with the 8.0 release, this setting is ignored, the modern format is preferred and the legacy format will be migrated to the modern format on upgrade.

7.48 `ssldbprefix` Option

The `ssldbprefix` base option specifies the prefixes of the certificate and key files.

7.49 `sslcompress` Option

The `sslcompress` base option determines whether support for the SSL/TLS Compression option ([RFC 3749](#)) is enabled. Enabling this is not recommended.

7.50 `sslpkix` Option

The `sslpkix` base option enables use of PKIX verification for SSL/TLS client certificates ([RFC 3280](#)). Full PKIX validation can involve network connections to validate certificates via OCSP or check CRLs. We have not tested these scenarios for correct operation when the system is under load and to verify that network timeouts and server shut down operate correctly when such verifications are in progress.

7.51 `sslrequiresafenegotiate` Option

Setting the `sslrequiresafenegotiate` base option requires all SSL/TLS peers to implement safe SSL re-negotiation as specified in [RFC 5746](#). In late 2009, an attack against the SSL/TLS protocol was discovered that makes any client that does not require secure re-negotiation insecure when talking to almost any SSL/TLS server that implements pre-5746 re-negotiation. While our servers are safe from the attack once the NSS 3.12.5 or later patch is installed, this option causes the server to refuse to talk to SSL/TLS clients unless those clients have also been upgraded to be safe from the attack. This feature can be helpful at a security-sensitive site to detect clients that need to be upgraded to improve site security.

7.52 `sslv3enable` Option

The `sslv3enable` base option determines whether legacy support for the SSLv3 protocol (as opposed to the modern TLS protocol) is enabled. This legacy support has been deprecated for some time so the default was changed from 1 to 0 in the 7.0.5.34.0 patch release.

7.53 `supportedlanguages` Option

The `supportedlanguages` base option specifies the languages supported by server code.

7.54 `threadholddelay` Option

The `threadholddelay` base option sets a thread hold delay time (in milliseconds) for IMAP and POP connections. This is the amount of time that asynchronous read and write operations will try to keep a worker thread around.

7.55 tmpdir Option Under base

The `tmpdir` base option specifies the temporary file directory; defaults to `$SERVERROOT/data/tmp/`.

On Linux, this option should instead be set to `/dev/shm/`.

7.56 ugldapbasedn Option

The `ugldapbasedn` base option specifies the root of the user/group configuration tree in the Directory Server. Normally this option is set to an appropriate value during initial configuration.

The MTA has a "twin" option, `ldap_user_root`, which may be used to override this base option for MTA-specific purposes.

7.57 ugldapbindcred Option

The `ugldapbindcred` base option specifies the password for the user/group administrator.

The MTA has a "twin" option, `ldap_password`, which may be used to override this base option for MTA-specific purposes.

7.58 ugldapbinddn Option

The `ugldapbinddn` base option specifies the DN of the user/group administrator. Normally initial configuration sets this option to a value of the form:

```
uid=msg-admin-<msg.ServerHostName>-<msg.product.InstallationTimestamp>, ou=People, <deforgdn>
```

The MTA has a "twin" option, `ldap_username`, which may be used to override this base option for MTA-specific purposes.

7.59 ugldaphost Option

The `ugldaphost` base option specifies the LDAP server list for user/group lookup. Takes a space-separated string. A port may be specified by appending `:port` to a host name in the list. If empty or not set, the loopback interface is used.

Normally this option is set to an appropriate value during initial configuration.

The MTA has a "twin" option, `ldap_host`, which may be set to specify an MTA-specific override of this option's value.

7.60 ugldapport Option

The `ugldapport` base option specifies the LDAP port for user/group lookup if a port is not specified in the `ugldaphost` list. The default is 389, though initial configuration may set it to a different value. As of the 7.0.5 release, if this is set to 636, then SSL will be used regardless of `ugldapusessl` setting.

The MTA has a "twin" option, `ldap_port`, which may be set to specify an MTA-specific override of this option's value.

7.61 ugldapusessl Option

The `ugldapusessl` base option if enabled says to use SSL to connect to the user/group LDAP server. Note that as of 7.0.5, if `ugldapport` is set to 636, then SSL will be used regardless of the value of `ugldapusessl`.

7.62 ldaprequiretls Option

If SSL is not already being used on a given LDAP connection (e.g., due to `ugldapusessl` or an `ldaps:` URL), enabling the `ldaprequiretls` base option will require successful negotiation of TLS (using LDAP StartTLS) before proceeding with the connection.

7.63 preferpoll Option

To improve performance, the IMAP and MMP servers use Solaris Event Completion Ports on Solaris instead of the poll system call starting with the Messaging Server 7.0.5 release. Since the Messaging Server 8.0.1 release, the servers use `epoll` on Linux instead of the poll system call. Setting the `preferpoll` option (available at base and MMP level) to 1 will revert to use of the standard Posix poll API instead. When `preferpoll` is set to 1, then the `polldelay` option also applies.

7.64 secret Option Under base

The `secret` [Base option](#) specifies a default to be used by the [Watcher](#), if the Watcher's own `watcher.secret` option has not been specified. The value should match that of the [Job Controller's secret](#), `job_controller.secret`.

7.65 rfc822headerallow8bit Option

The `rfc822headerallow8bit` base option, if set to 1, allow 8-bit characters in message headers in Messenger Express. If this parameter is set to 0 (in legacy configuration, a value of "no" for the `local.rfc822header.allow8bit configutil` parameter), or if the 8-bit character is invalid, then the character will be displayed as "?".

7.66 proxyadmin Option

The `proxyadmin` [base option](#) specifies the default store admin login name. It may be overridden for any particular backend host using the [imapadminproxy option](#).

7.67 proxyadminpass Option

The `proxyadminpass` [base option](#) specifies the default store admin password corresponding to the [proxyadmin](#) account. It may be overridden for each particular backend host using the [imapadminpassproxy option](#).

7.68 proxyimapport Option

The `proxyimapport` [base option](#) specifies the default IMAP port number for connections to backend store servers. It may be overridden for particular backend hosts by setting the `imapport` [proxy option](#) for that backend host, `proxy.hostname.imapport`, (where note that any periods in the hostname must be quoted at the `msconfig` command line using the backslash character).

7.69 proxyimapssl Option

The `proxyimapssl` [base option](#) enables SSL access to backend store servers. Defaults to 1 if the backend store IMAP port is 993, and 0 otherwise.

7.70 proxyserverlist Option

The `proxyserverlist` [base option](#) specifies a Message Store server list from which to list shared folders. Takes a space-separated string. Not configured by default.

7.71 proxytrustmailhost Option

The `proxytrustmailhost` [base option](#) controls whether to proxy commands such as `urlfetch` to the user's LDAP `mailHost`, if that server is not listed in [proxyserverlist](#).

7.72 dcroot Option

The `dcroot` [base option](#) specifies the root of the DC tree in Directory Server. Normally initial configuration sets this option to an appropriate value.

The MTA has a "twin" option, `ldap_domain_root`, which may be set to specify an MTA-specific override for this option.

7.73 dnsresolveclient Option

The `dnsresolveclient` [base option](#) forces servers -- the IMAP server, POP server, and MSHTTP server -- to perform a DNS reverse lookup on client connections to attempt to determine a corresponding host name. Note that if TCP wrappers are enabled -- see the [tcpaccess](#) server option -- then such DNS reverse lookups will be performed regardless of the setting of `dnsresolveclient`.

For the MMP and DNS reverse lookups, see the [clientlookup](#) option.

For the MTA's SMTP server and DNS reverse lookups, see the `ident*` channel options.

7.74 loginseparator Option

The `loginseparator` [base option](#) specifies the character(s) to be used as login separator (between `userid` and `domain`).

7.75 rbac Option

The `rbac` [base option](#) enables use of Role-Based Access Controls on Solaris (don't require root access).

7.76 listenaddr Option Under base

The `listenaddr` base option specifies the IPv4 address to listen on when accepting connections, or to bind to when making connections. The allowed values for the `listenaddr` option include an IPv4 address in dotted decimal form (e.g., 127.0.0.1), or a short form or fully-qualified DNS host name which will be resolved to an IPv4 address by obtaining the DNS A record for the name. To explicitly specify the default value of binding to all available interfaces, the string "INADDR_ANY" may be used. To bind to the loopback device, 127.0.0.1, the string "localhost" may be specified.

If `base.listenaddr` is not explicitly set, then the default for accepting connections is the string "INADDR_ANY", while the default for making connections is the loopback address.

Note that one of the (several) implications of `listenaddr` is that it sets the host IP for most servers in the product, including the ENS server. Versions prior to 7 Update 4 instead used a legacy syntax in the `local.ens.port` option for the ENS server host IP address.

7.77 sslnicknames Option Under base

List of SSL/TLS server certificate nicknames (only one per certificate type) to offer clients if SSL/TLS enabled. This base level list may be overridden for particular services.

7.78 welcomemsg Option Under base

The `welcomemsg` [Base option](#) specifies a welcome message for new users of the Message Store. The maximum size is 1 MB. Syntax: "\$" line separators, with headers.

7.79 softtokendir Option

When using both Sun Cluster and Solaris `libpkcs11` soft token, the Sun Cluster agent will clear the environment so the normal `SOFTTOKEN_DIR` environment variable can't be used. The `softtokendir` Base option will be used by the `ims_svc_*` utilities to set the `SOFTTOKEN_DIR` environment variable.

7.80 ipv6in Option

When set to a value of 1, the `ipv6in` option instructs Messaging Server to accept inbound IPv6 connections for all services provided that the host has at least one network interface configured for IPv6. Services specifically configured to listen on only IPv4 interfaces cannot also accept inbound IPv6 connections. When set to a value of 0, inbound IPv6 connections are not allowed.

Inbound IPv4 connections will always be permitted.

7.81 ipv6out Option

When set to a value of 1, the `ipv6out` option instructs Messaging Server to attempt outbound IPv6 connections for all services provided that the host has at least one network interface configured for IPv6. Services specifically configured to bind their source IP address only to IPv4 interfaces cannot attempt IPv6 outbound connections. For example, an SMTP client bound to a specific IPv4 interface cannot then establish an outbound IPv6 connection. When set to a value of 0, outbound IPv6 connections are not allowed.

When set to a value of 1, outbound services will attempt DNS lookups of both A and AAAA records. Connection attempts will then be made in the order dictated by the `ipv6sortorder` option. Note the DNS lookups will always request A records. This option only controls whether or not AAAA records are also requested.

7.82 `ipv6usegethostbyname` Option

By setting the `ipv6usegethostbyname` option to the value "1", Messaging Server will use `gethostbyname()` for all host name to IP address lookups. This has the immediate effect of forcing the use of IPv4 for all outbound connections and host name to IP address lookups. Usage of this option is restricted.

7.83 `ipv6sortorder` Option

The `ipv6sortorder` option controls the order in which IPv4 (A) and IPv6 (AAAA) DNS address records are used when attempting connections to other named systems.

Table 7.1 `ipv6sortorder` Option Values

Value	Behavior
default	Process A and AAAA records in the order returned by the operating system.
a	Process only A records; ignore AAAA records.
aaaa	Process only AAAA records; ignore A records.
a-aaaa	Process A records, then AAAA records.
aaaa-a	Process AAAA records, then A records

7.84 `projectid` Option Under `base`

The `projectid` option specifies the numeric identifier Messaging Server uses when obtaining shared memory segments. This identifier is used in `ftok()` calls to generate a shared memory segment key. By default, a value of 7 is used. Only the lowest eight bits of the value are significant.

7.85 `tlsv12enable` Option

The `tlsv12enable` base option determines whether TLS version 1.2 (the modern version of the SSL protocol) is enabled. For releases prior to 8.0 this defaulted to 0; starting with 8.0 this defaults to 1. As of MS 8.0.1, if `tlsminversion` is set to 1.2, this option is ignored and TLS 1.2 is enabled.

7.86 sslrenegotiate Option

The SSL/TLS protocol includes a re-negotiation feature that is primarily used by classic HTTP for client certificate authentication. This feature is not needed by Messaging Server and is disabled by default as of the 7.0.5.31.0 release. Setting this option will enable this feature.

7.87 logfile options

There are a number of options relating to nslog log files, set under a logfile group, under the appropriate component. logfile options may be set under [base](#), [http](#), [imap](#), [metermaid](#), [mmp](#), [mta](#), [imaproxy](#), [popproxy](#), [submitproxy](#), [messagetrace](#), [pop](#), [snmp](#), [tcp_lmtp_server](#), [msadmin](#), [ens](#), [job_controller](#), or [dispatcher](#).

For additional debugging, see also the [debugkeys](#) option available for a number of Messaging Server components.

Certain components support automatic log file rollover; see the [rollovermanager options](#).

7.87.1 expirytime Option

The `expirytime logfile` option, `component.logfile.expirytime`, specifies the maximum time in seconds a log file is kept. The default is 604800 seconds (corresponding to one week).

7.87.2 flushinterval Option

The `flushinterval logfile` option, `component.logfile.flushinterval`, specifies the time interval in seconds between logfile buffer flushes.

7.87.3 filemode Option

The `filemode logfile` option, `component.logfile.filemode`, specifies the file mode in octal used to create log files for the specified component. The value will be masked with octal 0666 and the process umask to set actual log file permissions. If you want a process with a different userid but in the same group as the Messaging Server user to have read access to log files, use 0640. The [store.umask](#) option can be used to modify the process umask to allow this.

Note that the `filemode` option does not apply to MTA debug logs or [MTA transaction log files](#).

For the 8.0 release and later, it is no longer necessary to modify the umask.

7.87.4 logmillisecond Option

When the `logmillisecond logfile` option, `component.logfile.logmillisecond`, is turned on (that is, set to 1), then milliseconds will be shown in nslog log files.

7.87.5 maxlogfiles Option

The `maxlogfiles logfile` option, `component.logfile.maxlogfiles`, specifies the maximum number of log files to retain.

7.87.5.1 Use with mmp Under logfile

For the MMP, the `maxlogfiles logfile` option, `mmp.logfile.maxlogfiles`, was new for Messaging Server 7u1.

7.87.6 maxlogfilesize Option

When an `nslog` log file for a component reaches the `component.logfile.maxlogfilesize` size, a log rollover operation will be triggered; logging may continue to the log file while the rollover operation is in progress. In the Messaging Server 7.0.5 release, the default value was increased to keep more historical data. In previous versions, the default value was 2097152 (2MB).

7.87.6.1 Use with mmp Under logfile

For the MMP, the `maxlogfilesize logfile` option, `mmp.logfile.maxlogfilesize`, was new for Messaging Server 7u1.

7.87.7 maxlogsize Option

When an `nslog` log file rollover operation occurs, if the maximum total size in bytes of all log files for this service exceeds the `component.logfile.maxlogsize` value, then the older log files will be removed as part of the rollover process until the sum of the file sizes observed at the start of the rollover operation falls below this threshold. In the Messaging Server 7.0.5 release, the default value was increased to keep more historical data. In previous versions, the default value was 20971520 (20MB).

7.87.7.1 Use with mmp Under logfile

For the MMP, the `maxlogsize logfile` option, `mmp.logfile.maxlogsize`, was new for Messaging Server 7u1.

7.87.8 rollovertime Option

The `rollovertime logfile` option, `component.logfile.rollovertime`, specifies the length of time in seconds to keep a log file active. That is, the maximum period of time to record data to a single log file. The default is 86400 seconds (corresponding to one day).

7.87.8.1 Use with mmp Under logfile

For the MMP, the `rollovertime logfile` option, `mmp.logfile.rollovertime`, was new for Messaging Server 7u1.

7.87.9 rolloverpolicy Option

The `rolloverpolicy logfile` option, `component.logfile.rolloverpolicy`, specifies the policy of rolling over an active log file. 0: disabled; 1: rollover based on time

specified by `rollovertime logfile` option; 2: rollover based on log file size specified by `maxlogfilesize logfile` option; 3: rollover based on both time and log file size.

7.87.10 syslogfacility Option

The `syslogfacility logfile` option, `component.logfile.syslogfacility`, specifies whether or not logging for that component is directed to the syslog service. The value of such an option can be `none`, `user`, `mail`, `daemon`, or `local0` to `local7`. If the value is set, messages are logged to the syslog facility corresponding to the set value and all other log file service options are ignored. The special value of `none` (which is the default) disables use of the syslog service.

7.87.10.1 Use with mmp Under logfile

For the MMP, the `syslogfacility logfile` option, `mmp.logfile.syslogfacility`, was new for Messaging Server 7u1.

7.88 debug Option

7.88.1 Use with deploymap

The debug Deployment Map option, `deploymap.debug`, enables the generation of debug output in the Deployment Map server or client's log file.

7.88.2 Use with msadmin

The `msadmin.debug` option enables debug output into the msadmin server's debug log file.

7.88.3 Use with pmxbl

The debug PureMessage IP Blocker option, (`pmxbl.debug`), if set to a non-zero value (maximum 999 -- attempting to set a larger value will result in the default value of 0 being used), enables debug output for the pmxbl callout routine.

7.88.4 Use with sms_gateway

The `sms_gateway.debugoption` enables debug output for the [SMS gateway](#). The default value is 6, which selects warning and error messages. The actual value of this option is a bit mask with the values shown below.

Table 7.2 SMS Gateway Debug Bit Mask Values

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.

7.88.5 Use with checkpoint

The debug Message Store checkpoint option (*i.e.*, `store.checkpoint.debug`), if set, enables stored checkpoint debug.

7.88.6 Use with domainmap

The `base.domainmap.debug` option sets the level of debugging messages for the domain map library code.

7.88.7 Use with job controller

The debug Job Controller option sets a bit mask for various types of debugging. When debugging is enabled, it is written to the Job Controller log file. That file is located in the MTA log directory, `SERVERROOT/log/`, and named `job_controller.log-uniqueid` where *uniqueid* is a unique name disambiguifying the file name. (Note that the [imsimta purge utility](#) understand the *uniqueids* and can be used to purge back older log files.)

7.88.8 Use with metermaid_client

The debug MeterMaid Client option enables debug output from the MTA client into SMTP log files.

7.88.9 Use with dispatcher

The debug Dispatcher option enables debug output from the Dispatcher.

7.89 Base autorestart options

Under the base group is the autorestart group, with merely two options available, `base.autorestart.enable` and `base.autorestart.timeout` (which may also be set and referred to simply as `autorestart.enable` and `autorestart.timeout`). Enabling autorestart means that Messaging Server can attempt to restart components that seem to be in trouble.

7.89.1 enable Option Under autorestart

The enable [Autorestart option](#) (`base.autorestart.enable` or more simply `autorestart.enable` in Unified Configuration, or `local.autorestart` in legacy configuration) enables automatic restart of failed or frozen (unresponsive) servers including IMAP, POP, HTTP, [Job Controller](#), [Dispatcher](#), and MMP servers.

7.89.2 timeout Option Under autorestart

The timeout [Autorestart option](#), (`autorestart.timeout` in Unified Configuration, or `local.autorestart.timeout` in legacy configuration), specifies a failure retry timeout. 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 the [crontab](#) option setting for `msprobe`'s schedule).

7.90 Base certmap options

Several options affect certificate map operation. These options are grouped under `base.certmap`, but may be referred to and set more simply as merely under `certmap`.

7.90.1 dncomps Option

The `dncomps` certmap option determines how to search for a client certificate in the directory. If this is not supplied, the server will expect the client certificate subject to contain the exact DN of the user in LDAP. If this is set to the empty string, a search will be performed under the [ugldapbasedn](#) based on a search filter from the [filtercomps](#). If this is set to a space-separated list of components, then a DN will be constructed by extracting the values of those components from the certificate subject in the order listed. If the component in the certificate subject is different from the component that will be used in the LDAP DN, then a translation is specified by using `subject-component=ldap-component` in the space separated list. For backwards compatibility, "mail" and "e" are treated as synonyms.

7.90.2 filtercomps Option

The `filtercomps` certmap option determines the search filter to use when locating the user entry in LDAP associated with a client certificate subject. If this is not provided or empty, then a search filter of `(objectclass=*)` will be used. If this is set to a space separated list of components, then a search filter will be formed by extracting values from components in the certificate subject in the order listed and combining them with a logical and. If the component in the certificate subject is different from the attribute name that will be used in the LDAP search filter, then a translation is specified by using `subject-component=ldap-component` in the space separated list. For backwards compatibility, "mail" and "e" are treated as synonyms.

Typically, `filtercomps` is not used unless [dncomps](#) is set to the empty string.

7.90.3 verifycert Option

If the `verifycert` certmap option is set to 1, then when a user record in LDAP is found for a given client certificate, the binary DER form of that certificate will be compared to the

`userCertificate;binary` attribute in LDAP. Authentication will succeed only if there is an exact match.

7.90.4 cmapldapattr Option

The `cmapldapattr certmap` option specifies the name of an LDAP attribute that will contain the certificate subjects for valid client certificates. There is no standard LDAP attribute defined for this purpose so it will be necessary to extend your LDAP schema. The attribute name `certSubjectDN` is suggested. If this is specified, the server will perform a subtree search under [ugldapbasedn](#) to locate a user. This is performed before the server attempts to use [dncomps](#) or [filtercomps](#) to find the user entry.

Chapter 8 Scheduler options

8.1 enable Option Under schedule	8-1
8.2 enablelog Option	8-1
8.3 Scheduler task options	8-1
8.3.1 enable Option Under task	8-2
8.3.2 crontab Option	8-2
8.3.3 return_job options	8-4
8.3.4 snapshot options	8-4
8.3.5 snapshotverify options	8-5

The Messaging Server Scheduler schedules and initiates execution of various periodic jobs for Messaging Server. These jobs may include the following named tasks:

- `return_job`, the MTA's message return job (message bouncer job), that returns (bounces) excessively old, undelivered messages,
- `expire`, the Message Store's message expiration job, that deletes *from disk* messages that users have deleted from their mailboxes,
- `msprobe`, checking whether Messaging Server components such as server processes are available ("up") and responsive,
- `purge`, the MTA's log file purge job, that purges "older" MTA log files,
- `snapshot`, the Message Store's database "snapshot" job, that captures a current-moment "snapshot" of the messages in the Message Store,
- `snapshotverify`, the Message Store's database "snapshot verification" job, that verifies whether snapshots are sound.

The only options for the Scheduler itself are `enable` (to enable the Scheduler's own operation) and `enablelog`. The settings of more interest are those under [named task groups under the Scheduler](#), configuring behavior of each named task.

8.1 enable Option Under schedule

The `enable` Scheduler option, `schedule.enable` (Unified Configuration) or `local.sched.enable` (legacy configuration), enables the Scheduler service on `start-msg` startup. This option defaults to 0 if not set, but initial configuration normally enables the option.

8.2 enablelog Option

To enable output from the Scheduler's tasks to go to log files in the `log` directory, set `enablelog` to 1.

8.3 Scheduler task options

When the [Scheduler](#) is enabled, each named task known to the Scheduler *may* be explicitly enabled or disabled via the task's own `enable` option (but typically the task's `enable` value

defaults appropriately based on which Messaging Server components are enabled), and if enabled (whether implicitly or explicitly) will be executed at the schedule set via the `crontab` option for that task; *e.g.*,

```
msconfig> show schedule.enable
role.schedule.enable = 1
msconfig> show mta.enable
role.mta.enable = 1
msconfig> show schedule.task:return_job.*
role.schedule.task:return_job.crontab = 30 0 * * * lib/return_job
```

The Scheduler supports the following named tasks in particular:

- `return_job`, the MTA's message return job (message bouncer job), that returns (bounces) excessively old, undelivered messages,
- `expire`, the Message Store's message expiration job, that deletes *from disk* messages that users have deleted from their mailboxes, and purges messages from users' mailboxes according to administrative criteria,
- `msprobe`, checking whether Messaging Server components such as server processes are available ("up") and responsive,
- `purge`, the MTA's log file purge job, that purges older MTA log files,
- `snapshot`, the Message Store's database "snapshot" job, that captures a current-moment "snapshot" of the messages in the Message Store,
- `snapshotverify`, the Message Store's database "snapshot verification" job, that verifies whether snapshots are sound.

Other tasks can be executed by the Scheduler. For instance, rather than executing the Message Store's `impurge` job as a daemon as normally configured when `store.enable` is enabled, a site can disable that daemon via `store.purge.enable=0` and instead configure the Scheduler to run the `impurge` command periodically.

8.3.1 enable Option Under task

The `enable` Scheduler task option, `schedule.task:name.enable` (Unified Configuration) or `local.schedule.name.enable` (legacy configuration), controls whether a task should be scheduled.

If the Scheduler has been enabled, `schedule.enable=1`, all tasks default to being scheduled, unless explicitly disabled via `schedule.task:task-name.enable=0`. That is, the purpose of the task-level `enable` option is to provide a way to *disable* tasks.

8.3.2 crontab Option

The `crontab` Scheduler task option sets a task run schedule, for a task enabled with `schedule.task:task-name.enable` (Unified Configuration) or `local.schedule.*.enable` (legacy configuration). The `schedule.task:task-`

`name.crontab` option uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

8.3.2.1 Use with expire Under task

The crontab Scheduler task option for the `expire` task controls the interval for running `imexpire`, enabled with `schedule.task:expire.enable` (Unified Configuration) which defaults to the setting of the `store.enable` setting, (or in legacy configuration, `local.schedule.expire.enable` which defaults to the setting of `local.store.enable`). `schedule.task:expire.crontab` uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration normally sets this to:

```
msconfig> show schedule.task:expire.crontab
role.schedule.task:expire.crontab = 0 23 * * * bin/imexpire
```

8.3.2.2 Use with msprobe Under task

The crontab Scheduler task option for the `msprobe` task controls the `msprobe` run schedule, enabled with `schedule.task:msprobe.enable` (Unified Configuration) or `local.schedule.msprobe.enable` (legacy configuration). `msprobe` is a daemon that probes servers to see if they respond to service requests. `schedule.task:msprobe.crontab` uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration sets this to:

```
msconfig> show schedule.task:msprobe.crontab
role.schedule.task:msprobe.crontab = 5,15,25,35,45,55 * * * * lib/msprobe
```

8.3.2.3 Use with purge Under task

The crontab Scheduler task option for the `purge` task controls the interval for running `imsimta purge`, enabled with `schedule.task:purge.enable` (Unified Configuration) which defaults to the value of `mta.enable`, (or in legacy configuration `local.schedule.purge.enable` which defaults to the value of `local.imta.enable`). `imsimta purge` removes older MTA log files. `schedule.task:purge.crontab` uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration sets this to:

```
msconfig> show schedule.task:purge.crontab
role.schedule.task:purge.crontab = 0 0,4,8,12,16,20 * * * bin/imsimta purge -num=5
```

8.3.2.4 Use with return_job Under task

Interval for running `return_job`, enabled with `schedule.task:return_job.enable` (Unified Configuration) which defaults to the setting of `mta.enable` (or in legacy configuration `local.schedule.return_job.enable` which defaults to the setting of

local.imta.enable). schedule.task:return_job.crontab uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

8.3.2.5 Use with snapshot Under task

The imdbverify snapshot and verify option, enabled with `schedule.task:snapshot.enable` (Unified Configuration) which defaults to `store.enable` (or in legacy configuration `local.schedule.snapshot.enable` which defaults to `local.store.enable`). imdbverify will take a snapshot backup copy of the database and verify it for use during automatic recovery.

Initial configuration sets this to:

```
msconfig> show schedule.task:snapshot.crontab
role.schedule.task:snapshot.crontab = 0 2 * * * bin/imdbverify -s -m
```

8.3.2.6 Use with snapshotverify Under task

The imdbverify utility updates snapshots of the mboxlist database incrementally. The snapshots can be used during automatic recovery. The imdbverify utility is enabled with `schedule.task:snapshotverify.enable` (Unified Configuration) or `local.schedule.snapshotverify.enable` (legacy configuration).

Initial configuration sets this to:

```
msconfig> show schedule.task:snapshotverify.crontab
role.schedule.task:snapshotverify.crontab = 5,15,25,35,45,55 * * * * bin/imdbverify
```

8.3.3 return_job options

The return_job [Scheduler task](#) has a couple of options.

8.3.3.1 enable Option Use With return_job Under task

The enable Scheduler task option for the return_job task controls whether the return_job task should be scheduled. Defaults to the setting of the `mta.enable` option (Unified Configuration) or `local.imta.enable` configutil parameter (legacy configuration). Starting with the 8.0 release, the `mta.enable` option is deprecated so this instead defaults to the value of the `job_controller.enable` option (Unified Configuration) or the `local.job_controller.enable` configutil parameter (legacy configuration). As the `job_controller.enable` option defaults to the value of the `mta.enable` option, upgrading customers should see no behavior change.

8.3.3.2 crontab Option Use With return_job Under task

Interval for running return_job, enabled with `schedule.task:return_job.enable` (Unified Configuration) which defaults to the setting of `mta.enable` (or in legacy configuration `local.schedule.return_job.enable` which defaults to the setting of `local.imta.enable`). schedule.task:return_job.crontab uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

8.3.4 snapshot options

See also the [Scheduler's snapshotverify task options](#). See also the [snapshotdirs](#) and [snapshotpath](#) Message Store options.

8.3.4.1 enable Option Use With snapshot Under task

The `enable` Scheduler task option for the `snapshot` task controls whether the snapshot / verify task should be scheduled. Defaults to the setting of the `store.enable` option (Unified Configuration) or `local.store.enable` `configutil` parameter (legacy configuration).

8.3.4.2 crontab Option Use With snapshot Under task

The `imdbverify` snapshot and verify option, enabled with `schedule.task:snapshot.enable` (Unified Configuration) which defaults to `store.enable` (or in legacy configuration `local.schedule.snapshot.enable` which defaults to `local.store.enable`). `imdbverify` will take a snapshot backup copy of the database and verify it for use during automatic recovery.

Initial configuration sets this to:

```
msconfig> show schedule.task:snapshot.crontab
role.schedule.task:snapshot.crontab = 0 2 * * * bin/imdbverify -s -m
```

8.3.5 snapshotverify options

See also the [Scheduler snapshot task options](#).

8.3.5.1 enable Option Use With snapshotverify Under task

The `enable` Scheduler task option for the `expire` task controls whether the process log verify task for rolling backups should be scheduled. Defaults to the setting of the `store.enable` option (Unified Configuration) or `local.store.enable` `configutil` parameter.

8.3.5.2 crontab Option Use With snapshotverify Under task

The `imdbverify` utility updates snapshots of the `mboxlist` database incrementally. The snapshots can be used during automatic recovery. The `imdbverify` utility is enabled with `schedule.task:snapshotverify.enable` (Unified Configuration) or `local.schedule.snapshotverify.enable` (legacy configuration).

Initial configuration sets this to:

```
msconfig> show schedule.task:snapshotverify.crontab
role.schedule.task:snapshotverify.crontab = 5,15,25,35,45,55 * * * * bin/imdbverify
```



Chapter 9 Watcher options

9.1 <code>enable</code> Option Under <code>watcher</code>	9-1
9.2 <code>port</code> Option Under <code>watcher</code>	9-1
9.3 <code>secret</code> Option Under <code>watcher</code>	9-1

The Watcher has just a few options.

9.1 `enable` Option Under `watcher`

The `enable` Watcher option, `watcher.enable` (Unified Configuration) or `local.watcher.enable` (legacy configuration), enables the Watcher service on `start-msg` startup. The Watcher service is a daemon that monitors Messaging Server and restarts services that fail. Refer to [`autorestart.enable`](#) (`local.autorestart` in legacy configuration) and the Administration Guide for details.

This option defaults to 0 if not set, but initial configuration normally enables the option.

9.2 `port` Option Under `watcher`

The `port` Watcher option specifies the `watcher` listen port.

9.3 `secret` Option Under `watcher`

The `secret` Watcher option specifies the shared secret used by the Watcher when communicating with watched processes. If `watcher.secret` is not specified, it will default to the value of the secret [Base option](#), `base.secret`. The value should match that of the [Job Controller's](#) secret, `job_controller.secret`.



Chapter 10 msprobe options

10.1 enable Option Use With msprobe Under task	10-1
10.2 queuedir Option	10-1
10.3 timeout Option Under msprobe	10-1
10.4 warningthreshold Option Under msprobe	10-1
10.5 probe options	10-2
10.6 crontab Option Use With msprobe Under task	10-2

There are a few options affecting msprobe operation. msprobe's service-specific [probes](#) can also individually override some of the general msprobe values.

When msprobe detects a possible problem, it can, depending upon other configuration, potentially let the Watcher know (at which point the Watcher can attempt to restart a troubled component) and/or generate an alarm message; see the [Watcher options](#) and [Alarm options](#), respectively.

See also the [stressperiod](#) and [stressfdwait](#) [base options](#) (which currently affect only the MMP).

10.1 enable Option Use With msprobe Under task

The `enable` Scheduler task option for the `msprobe` task controls whether the `msprobe` task should be scheduled.

10.2 queuedir Option

The `queuedir` msprobe option specifies the full pathname of spool directory or local queue directory to be monitored by `msprobe`. On an MTA system, to have `msprobe` monitor the MTA queue area, set this option to `DATAROOT/queue/`; since this option has no default value, leaving it unset means that `msprobe` will not monitor the MTA queue area.

10.3 timeout Option Under msprobe

The `msprobe.timeout` msprobe option specifies the time in seconds that `msprobe` waits after sending a request that goes unfulfilled before restarting a service. This is a general default for `msprobe`'s probes; a service-specific probe can set its own, override timeout via the [msprobe.probe:service-name.timeout](#) option.

Attempting to set a value of 0 will result in the value 30 (the default) getting used as the `msprobe` default.

10.4 warningthreshold Option Under msprobe

The `msprobe.warningthreshold` option sets a default warning threshold for any `msprobe probe` that does not have its own, more explicit, warning threshold set (via a `msprobe.probe:name.warningthreshold` option).

10.5 probe options

`msprobe`'s service-specific probes can set their own `warningthreshold` and `timeout` values, overriding `msprobe's general default such values`. Such probe values are set within a named probe group, where the name may (currently) be one of:

- `cert`
- `deploymap`
- `ens`
- `http`
- `imap`
- `job_controller`
- `lntp`
- `metermaid`
- `pop`
- `smtp`
- `submit`

So for instance:

```
msconfig> set msprobe.probe:submit.timeout 120
```

10.6 crontab Option Use With `msprobe` Under task

The `crontab Scheduler task` option for the `msprobe` task controls the `msprobe` run schedule, enabled with `schedule.task:msprobe.enable` (Unified Configuration) or `local.schedule.msprobe.enable` (legacy configuration). `msprobe` is a daemon that probes servers to see if they respond to service requests. `schedule.task:msprobe.crontab` uses UNIX `crontab` format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration sets this to:

```
msconfig> show schedule.task:msprobe.crontab
role.schedule.task:msprobe.crontab = 5,15,25,35,45,55 * * * * lib/msprobe
```

Chapter 11 Alarm options

11.1 noticehost Option	11-1
11.2 noticeport Option	11-1
11.3 noticercpt Option	11-1
11.4 noticesender Option	11-1
11.5 noticetemplate Option	11-2
11.6 Alarm system options	11-2
11.6.1 description Option Use With diskavail Under system	11-2
11.6.2 description Option Use With serverresponse Under system	11-2
11.6.3 statinterval Option	11-2
11.6.4 threshold Option	11-3
11.6.5 thresholddirection Option	11-3
11.6.6 warninginterval Option	11-3

msprobe can generate [warning messages](#) (so-called "alarms") if it detects possible problems such as server non-responsiveness, or disk unavailability. Several options control such warning ("alarm") messages. The options relating to the submission and format of such messages are set directly under the top-level `alarm` group. Options relating to the triggering of generation of such warning ("alarm") messages are set under a named `alarm.system` group.

11.1 noticehost Option

The `noticehost` alarm option specifies the SMTP host to which `msprobe` should submit warning messages. Often this would be the same host on which `msprobe` is running, and thus have the same value as the `smtphost` MSHTTP option (which itself defaults, if not set, to the value of the `hostname` Base option). However, if `msprobe` is running on a system with no SMTP server, a system which instead is an [LMTP back end system](#), then `noticehost` should be set explicitly to the name of an SMTP host. If `noticehost` is not set, it will default (as of Messaging Server 7 update 2) to the value of `http.smtphost` (in legacy configuration, `service.http.smtphost`), or the loopback address.

11.2 noticeport Option

The `noticeport` alarm option specifies the SMTP port to which `msprobe` will connect when submitting alarm messages.

11.3 noticercpt Option

The `noticercpt` alarm option specifies the recipient of `msprobe` alarm messages. If not set, "Postmaster@<local-hostname>" will be used, where `local-hostname` is the value of the `hostname` Base option.

11.4 noticesender Option

The `noticesender` Alarm option specifies the address used in From: header (and envelope From) of `msprobe` alarm messages. If not set, "Postmaster@<local-hostname>" will be used, where `local-hostname` is the value of the `hostname` Base option.

11.5 noticetemplate Option

The `noticetemplate` alarm option specifies the `msprobe` alarm message template. `%s` in the template is replaced with the following in order: sender, recipient, alarm description, alarm instance, alarm current value and alarm summary text. DELETED: Too prone to format errors; support dropped in 6.3 release; use of the former default value is now hard-coded.

As of JES MS 6.3, and the removal of this option, the `msprobe` alarm messages are always constructed as:

```
From: <noticesender>
To: <noticercpt>
Subject: ALARM: <description> of "<instance>" is <current-value>

<summary-text>
```

(corresponding to the former default template).

11.6 Alarm system options

Options regarding the triggering of `msprobe` warning ("alarm") messages regarding various components or "systems" are set under `alarm.system:system-name`, where `system-name` is either `diskavail` or `serverresponse`.

11.6.1 description Option Use With diskavail Under system

The `alarm.system:diskavail.description` option specifies the description for the `diskavail` alarm.

11.6.2 description Option Use With serverresponse Under system

The `alarm.system:serverresponse.description` option specifies the description for the `serverresponse` alarm.

11.6.3 statinterval Option

The `statinterval` option under a named `alarm.system` group, so either `alarm.system:diskavail.statinterval` or `alarm.system:serverresponse.statinterval`, specifies the interval in seconds between checks on the named system. Set to 0 to disable checks.

11.6.3.1 Use with diskavail Under system

The `alarm.system:diskavail.statinterval` option specifies the interval in seconds between disk availability checks. Set to 0 to disable checks of disk usage.

11.6.3.2 Use with serverresponse Under system

The `alarm.system:serverresponse.statinterval` option specifies the interval in seconds between checks on server responsiveness. Set to 0 to disable checking of server response.

11.6.4 threshold Option

Description of the threshold measurement.

11.6.4.1 Use with `diskavail` Under system

The `alarm.system:diskavail.threshold` option specifies the percentage of disk space availability below which an alarm is sent.

11.6.4.2 Use with `serverresponse` Under system

The `alarm.system:serverresponse.threshold` option specifies the server response time, in seconds, triggering an alarm.

11.6.5 thresholddirection Option

Specifies whether an alarm is issued when the measurement is greater than (1) or less than (-1) the specified threshold.

11.6.5.1 Use with `diskavail` Under system

The `alarm.system:serverresponse.thresholddirection` option specifies whether the alarm is issued when disk space availability is below threshold (-1) or above it (1).

11.6.5.2 Use with `serverresponse` Under system

The `alarm.system:serverresponse.thesholddirection` option specifies whether an alarm is issued when server response time is greater than (1) or less than (-1) the threshold.

11.6.6 warninginterval Option

The `warninginterval` Alarm system option specifies, for a named system, the interval in hours between subsequent issuances of the alarm.

11.6.6.1 Use with `diskavail` Under system

`msprobe` can generate warning messages if it detects possible disk availability problems. The `alarm.system:diskavail.warninginterval` option specifies the interval in hours between subsequent repetition of disk availability alarms.

11.6.6.2 Use with `serverresponse` Under system

`msprobe` can generate warning messages if it detects possible server problems (*i.e.*, server non-responsiveness). The `alarm.system:serverresponse.warninginterval` option specifies the interval in hours between subsequent repetition of server response alarm.



Chapter 12 Auth options

12.1 auto_transition Option	12-1
12.2 usedomainmap Option	12-1
12.3 has_plain_passwords Option	12-1
12.4 searchfilter Option	12-1
12.5 canonicalsearchfilter Option	12-1
12.6 searchfordomain Option	12-1
12.7 broken_client_login_charset Option	12-2

A number of options may be set under the auth group to affect authentication in general.

12.1 auto_transition Option

When the `auto_transition` Auth option is set to 1 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 or CRAM-MD5.

12.2 usedomainmap Option

The `usedomainmap` auth option controls whether to look up domains prior to locating users when performing authentication. If disabled, then search the entire user/group subtree when authenticating a user.

12.3 has_plain_passwords Option

The `has_plain_passwords` Auth option is a boolean to indicate that the directory stores plaintext passwords, which enables APOP and CRAM-MD5.

12.4 searchfilter Option

The `searchfilter` Auth option specifies the default search filter used to look up users for basic authentication and identity purposes when one is not specified in the `inetDomainSearchFilter` for the domain. The syntax is the same as `inetDomainSearchFilter` (see schema guide).

12.5 canonicalsearchfilter Option

The `canonicalsearchfilter` Auth option value is used when locating a user in an LDAP domain using the user's canonical identity. When a user authenticates, a translation is done from authentication identity to canonical identity. With default settings there is no difference between these two identities and the search filters are the same. However, if a site wishes to have users authenticate using an attribute other than `uid`, then these identities can be different and thus different search filters are needed for authentication user lookup and canonical user lookup. The syntax is the same as `inetDomainSearchFilter` (see schema guide).

12.6 searchfordomain Option

By default, the authentication system looks up the domain in LDAP following the rules for domain lookup, and 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 [searchfilter](#) option) occurs directly under the LDAP tree specified by the [ugldapbasedn](#) option. 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.

12.7 broken_client_login_charset Option

Some mail clients violate the IMAP and POP standards that require usernames and passwords to be in US-ASCII for the LOGIN and USER/PASS commands. These broken clients may instead use another charset, such as UTF-8 or ISO-8859-1 for usernames and passwords. Note that standards compliant clients may use the SASL PLAIN mechanism for IMAP, POP and SMTP submission. SASL PLAIN requires use of the UTF-8 charset and thus supports multiple languages with interoperable codepoints.

When the `broken_client_login_charset` Auth option has the default value of UTF-8, clients that incorrectly send UTF-8 for the LOGIN or USER/PASS commands will be allowed to authenticate and will interoperate with clients that use standards-compliant SASL PLAIN usernames and passwords.

When this option is set to the ISO-8859-1 value, then a three step workaround is enabled to attempt to achieve partially interoperable behavior:

1. First, the usernames and passwords are converted from ISO-8859-1 to UTF-8 and a standards-compliant search and bind to the LDAP directory is attempted. If this succeeds, the user is authenticated and everything works fine. If the search fails, then the username does not exist in the directory and the authentication fails.
2. If the standard bind fails, Messaging Server will attempt to use the ISO-8859-1 password in an LDAP simple bind operation (this step is compliant with the 1997 version of LDAP, but not the 2006 version of LDAP). Directory Server Enterprise Edition does not enforce the UTF-8 password restriction for simple bind and is thus compatible with this second step of the workaround. If this succeeds, the user is considered authenticated.
3. After a successful non-standard LDAP bind, Messaging Server will attempt to correct the in-compliant password entry in the LDAP directory by writing the UTF-8 version of the password to the user's `userPassword` attribute. If this succeeds, subsequent authentications for that user will be faster and standards compliant and the user will be compatible with standard authentication mechanisms such as SASL PLAIN. When this step occurs a message is written to the log at `notice` log level.

Before setting this to a non-default value, customers should verify they have no other systems that perform LDAP simple bind operations with a charset other than UTF-8 to the LDAP server used by Messaging Server. LDAP clients that violate [RFC 4511](#) in that way will not interoperate with standard use of SASL PLAIN or this workaround.

For step 3 to operate correctly, the Messaging Server End User administrator (as specified in the [ugldapbinddn option](#)) must have write access to the `userPassword` attribute. The LDAP Access Control Instructions (ACI) set up by Messaging Server's `configure` utility do not include this write access, so the LDAP ACI titled `Messaging Server End User Administrator Write Access Rights` on the `user/group` tree must be updated to add

userPassword to the attribute list. See the Directory Server documentation for instructions on editing Directory Server Access Control.

Chapter 13 sectoken options

13.1 tokenpass Option	13-1
-----------------------------	------

The only security token option is `tokenpass`, which may be set inside a named `sectoken` group, taking the place in Unified Configuration of settings made in the `sslpassword.conf` file in legacy configuration.

13.1 tokenpass Option

The `tokenpass` option is set inside a named `sectoken` group; this Unified Configuration setting takes the place of what in legacy configuration would be set as a `token-name:password` pair inside the `sslpassword.conf` file.

When configuring Secure Sockets Layer (SSL) or Transport Layer Security (TLS), the private keys are stored in a configuration private key file (presently `key8.db`) or a physical hardware device. These storage locations are collectively referred to as tokens. The private keys are typically encrypted with a password that the server requires to accept an SSL or TLS connection. The password for a given token is stored in the `tokenpass` option in a `sectoken` group whose name is the name of the token. The default software token's name is "Internal (Software) Token".

For instance, the legacy configuration setting in the `sslpassword.conf` file of:

```
Internal (Software) Token:whatever
```

would be set using Unified Configuration as:

```
msconfig> set "sectoken:Internal (Software) Token.tokenpass" whatever
```



Chapter 14 Deployment Map options

14.1 enable Option Under deploymap	14-1
14.2 debug Option Under deploymap	14-1
14.3 heartbeat Option	14-1
14.4 port Option Under deploymap	14-1
14.5 server_host Option Under deploymap	14-1
14.6 run_as_server Option	14-2
14.7 userid Option	14-2
14.8 capability_starttls Option Under deploymap	14-2

Several options relate to the Deployment Map service.

Note that msprobe can probe for whether the Deployment Map server is running; see msprobe's [probe options](#).

14.1 enable Option Under deploymap

The enable [Deployment Map option](#), `deploymap.enable`, enables use of the Deployment Map service. Whether to run as a client or server is specified with the `run_as_server` option. The default is to run as a client.

14.2 debug Option Under deploymap

The debug Deployment Map option, `deploymap.debug`, enables the generation of debug output in the Deployment Map server or client's log file.

14.3 heartbeat Option

Deployment Map clients keep their TCP connection open indefinitely to the Deployment Map server. Typically, there is only traffic over the connection when a client connects or disconnects or a change is made to the deployment map. Consequently, the TCP connection can remain silent for hours if not days. To prevent network hardware from closing the connection due to inactivity, the client periodically sends a simple heartbeat to the server. By default, this heartbeat is sent every 30 minutes plus or minus a random number of seconds. The period of this heartbeat is controlled with this option. To disable heartbeats entirely, specify a value of 0.

14.4 port Option Under deploymap

The port Deployment Map option, `deploymap.port`, specifies the TCP port on which the Deployment Map service listens for incoming TCP connections.

14.5 server_host Option Under deploymap

The `server_host` Deployment Map option specifies the fully qualified hostname or IP address of the remote Deployment Map server. This option is only used when the `run_as_server` option is set to 0 (false) or not specified. Use the `port` option to specify the TCP port which the remote server listens on.

14.6 run_as_server Option

The Deployment Map service consists of a single server running on one Messaging Server host and all other Messaging Server hosts running Deployment Map clients. When the Deployment Map service is enabled, the host runs a Deployment Map client by default. To instead run a Deployment Map server, set this option to the value 1. The other hosts -- the clients -- should either not specify this option or, if they do specify it, set it to the value 0. Further, clients must specify the `server_host` option (`local.deploymap.serverhost` for legacy config). That option specifies the hostname or IP address of the host running the Deployment Map server.

14.7 userid Option

The `userid` and `passwd` options (`local.deploymap.userid` and `local.deploymap.passwd` for legacy config) are the shared secret pair used by clients to authenticate with the server. They will be sent in the clear unless the use of SSL has been enabled with the `deploymap.sslusessl` option (`local.deploymap.sslusessl` in legacy configuration).

14.8 capability_starttls Option Under deploymap

The `capability_special_use` `deploymap` option, when set to 1 (the default), causes the Deployment Map server to advertise the STARTTLS extension. Advertising TLS (SSL) support does not mandate its use. Requiring use of TLS (SSL) is accomplished with the `deploymap.sslusessl` option.

Chapter 15 rollovermanager options

15.1 enable Option Under rollovermanager 15-1

The only rollovermanager option is `rollovermanager.enable`.

15.1 enable Option Under rollovermanager

The enable rollovermanager option enables its operation.



Part III The Message Store

Messaging Server provides a Message Store for storing users' email, and provides servers that support email client access to user mail.

There are a great many options for controlling and modifying Message Store operation; besides general [Base options](#), see specifically the [Message Store options](#) and [Partition options](#). A few options relating to localization of automatically generated messages may be found under [message_language_options](#).

For configuration of the servers that support email client access to user mail in the Message Store, see [IMAP options](#), [POP options](#), and [MSHTTP options](#).



Chapter 16 Message Store options

16.1 enable Option Under store	16-4
16.2 backupdir Option	16-4
16.3 cachesynclevel Option	16-5
16.4 checkdiskusage Option	16-5
16.5 checkmailhost Option	16-5
16.6 dbnumcaches Option	16-5
16.7 dbsync Option	16-5
16.8 deadlockaggressive Option	16-5
16.9 diskusagethreshold Option	16-5
16.10 ensureownerrights Option	16-6
16.11 expungesynclevel Option	16-6
16.12 finalcheckpoint Option	16-6
16.13 indexsynclevel Option	16-6
16.14 listimplicit Option	16-6
16.15 logexpungedetails Option	16-6
16.16 maxfolders Option	16-6
16.17 maxlog Option	16-6
16.18 maxmessages Option	16-7
16.19 messagesynclevel Option	16-7
16.20 overquotastatus Option	16-7
16.21 perusersynclevel Option	16-7
16.22 pin Option	16-7
16.23 quotaoverdraft Option	16-8
16.24 rollingdbbackup Option	16-8
16.25 seenckpinterval Option	16-8
16.26 seenckpstart Option	16-8
16.27 sharedfolders Option	16-8
16.28 snapshotdirs Option	16-8
16.29 snapshotpath Option	16-8
16.30 subscribesynclevel Option	16-9
16.31 synclevel Option	16-9
16.32 backupexclude Option	16-9
16.33 admins Option	16-9
16.34 indexeradmins Option	16-9
16.35 autorepair Option	16-9
16.36 autorepairdebug Option	16-10
16.37 cleanupage Option	16-10
16.38 cleanupsize Option	16-10
16.39 dbcachesize Option Under store	16-10
16.40 dblogregionmax Option	16-10
16.41 dbregionmax Option	16-10
16.42 folderlockcount Option	16-11
16.43 dbtmpdir Option	16-11
16.44 defaultmailboxquota Option	16-11
16.45 defaultmessagequota Option	16-11
16.46 defaultpartition Option	16-11
16.47 maxcachefilesize Option	16-11
16.48 cachepreviewlen Option	16-12
16.49 mailboxpurgedelay Option	16-12
16.50 encryptnew Option	16-12

16.51	keypass Option	16-12
16.52	keylabel Option	16-12
16.53	quotaenforcement Option	16-12
16.54	quotaexceededmsginterval Option	16-12
16.55	quotagraceperiod Option	16-12
16.56	quotanotification Option	16-12
16.57	quotawarn Option	16-13
16.58	serviceadmingroupdn Option	16-13
16.59	umask Option	16-13
16.60	expiresieve Option	16-13
16.61	quotaexceededmsg Option Under store	16-13
16.62	Message Store archive options	16-13
16.62.1	tmpdir Option Under archive	16-14
16.62.2	compliance Option	16-14
16.62.3	operational Option	16-14
16.62.4	source_channel Option	16-14
16.62.5	destination Option	16-14
16.62.6	style Option	16-14
16.62.7	reportdir Option	16-15
16.62.8	intext Option	16-15
16.62.9	posteditmode Option	16-15
16.62.10	useheaderrecipients Option	16-15
16.62.11	retrieveport Option	16-15
16.62.12	retrieveserver Option	16-15
16.62.13	retrievetimeout Option	16-15
16.62.14	path Option Under archive	16-15
16.63	Message Store checkpoint options	16-16
16.63.1	stresslimit Option	16-16
16.63.2	debug Option Under checkpoint	16-16
16.64	Message Store dbreplicate options	16-16
16.64.1	enable Option Under dbreplicate	16-16
16.64.2	port Option Under dbreplicate	16-16
16.64.3	dbremotehost Option	16-16
16.64.4	dbpriority Option	16-16
16.64.5	twosites Option	16-17
16.64.6	queuemax Option	16-17
16.64.7	ackpolicy Option	16-17
16.64.8	acktimeout Option	16-17
16.65	Message Store deadlock options	16-17
16.65.1	autodetect Option	16-17
16.65.2	checkinterval Option	16-17
16.66	Message Store expire options	16-18
16.66.1	exploglevel Option	16-18
16.66.2	crontab Option Use With expire Under task	16-18
16.67	Message Store expirerule options	16-18
16.67.1	deleted Option	16-19
16.67.2	exclusive Option	16-19
16.67.3	folderpattern Option	16-19
16.67.4	foldersizebytes Option	16-19
16.67.5	messagecount Option	16-19
16.67.6	messagedays Option	16-19
16.67.7	messagesize Option	16-19
16.67.8	messagesizedays Option	16-19

16.67.9	seen Option	16-20
16.68	Message Store folderquota options	16-20
16.68.1	enable Option Under folderquota	16-20
16.69	Message Store messagetype and typequota options	16-20
16.69.1	enable Option Under messagetype	16-21
16.69.2	enable Option Under typequota	16-21
16.69.3	header Option	16-21
16.69.4	contenttype Option	16-21
16.69.5	flagname Option	16-22
16.69.6	quotaroot Option	16-22
16.70	Message Store msghash options	16-22
16.70.1	enable Option Under msghash	16-22
16.70.2	dbcachesize Option Under msghash	16-22
16.70.3	nummsgs Option	16-22
16.71	Message Store purge options	16-22
16.71.1	enable Option Under purge	16-23
16.71.2	count Option	16-23
16.71.3	maxthreads Option Under purge	16-23
16.71.4	percentage Option	16-23
16.71.5	crontab Option Use With purge Under task	16-24
16.72	Message Store relinker options	16-24
16.72.1	enable Option Under relinker	16-24
16.72.2	maxage Option	16-24
16.72.3	minsize Option	16-24
16.72.4	purgecycle Option	16-25
16.73	Message Store shared folder options	16-25
16.73.1	restrictanyone Option	16-25
16.73.2	restrictdomain Option	16-25
16.73.3	shareflags Option	16-25
16.73.4	user Option Under publicsharedfolders	16-25
16.74	Message Store deleted options	16-26
16.74.1	backupbadidxfiles Option	16-26
16.74.2	listrecover Option	16-26
16.74.3	local.store.maxlogs Option	16-26
16.74.4	messagetypeplugin Option	16-26
16.74.5	local.store.serversidewastebasket Option	16-26
16.74.6	snapshotinterval Option	16-26
16.74.7	defaultacl Option	16-26
16.74.8	diskflushinterval Option	16-26
16.74.9	expirestart Option	16-26
16.74.10	mailboxexpungesize Option	16-26
16.74.11	cleanonly Option	16-27
16.74.12	workday Option	16-27

Many options for the Message Store are set directly under the store group, *e.g.*:

```
msconfig> set store.option-name option-value
```

Under the store group are also other named groups with their own additional options:

- [archive](#)
- [checkpoint](#)

- [dbreplicate](#)
- [deadlock](#)
- [expire](#)
- [expirerule](#)
- [folderquota](#)
- [messagetype](#)
- [msghash](#)
- [privatesharedfolders](#)
- [publicsharedfolders](#)
- [purge](#)
- [relinker](#)
- [typequota](#)

Options under a store subgroup may be set with a command of the form:

```
msconfig> set store.subgroup.option-name option-value
```

e.g.,

```
msconfig> set store.deadlock.checkinterval 12
```

A number of Message Store options from previous versions are now obsolete and have been [deleted](#).

Note that options specifically about Message Store partition setup are grouped under the (top-level) [partition](#) group; they are not grouped under `store`.

Of course [base level options](#), as they affect Messaging Server as a whole, can be significant for Message Store operation. But there are a couple of base level options that are particularly oriented towards the Message Store:

- [dbtxnsync](#)
- [enablelastaccess](#)
- [dblockcount](#)
- [welcomemsg](#)

16.1 enable Option Under store

The enable Message Store option, `store.enable` (Unified Configuration) or `local.store.enable` (legacy configuration), enables the Message Store when starting services. This option defaults to 0 if not set, but initial configuration normally enables the option.

16.2 backupdir Option

The `backupdir` Message Store option specifies the directory for backup image of Message Store data.

16.3 cachesynclevel Option

The `cachesynclevel` Message Store option controls the synchronization level for store cache file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `cachesynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `cachesynclevel`.

16.4 checkdiskusage Option

The `checkdiskusage` Message Store option enables stopping messages from being delivered to a Message Store partition when the partition fills more than a [specified percentage](#) of available disk space. If disk usage goes higher than the specified threshold, the store daemon locks the partition and logs a message to the default log files. When disk usage falls below the threshold, the partition is unlocked, and messages are again delivered to the store.

See also the `alarm.system:diskavail.threshold` option which sets a threshold for disk availability below which an alarm message will be sent to the postmaster.

16.5 checkmailhost Option

The `checkmailhost` Message Store option enables checking that the user `mailhost` attribute matches this server.

16.6 dbnumcaches Option

The `dbnumcaches` Message Store option controls the number of mboxlist db caches. If `dbnumcaches` is 0 or 1, the cache will be allocated contiguously in memory. If it is `n` greater than 1, the cache will be broken up into `n` equally sized, separate pieces of memory. The maximum value permitted is 32.

16.7 dbsync Option

The `dbsync` Message Store option affects flushing of store database data to disk. If this is set to 1, cached database information will be flushed to disk before the database file is closed.

16.8 deadlockaggressive Option

A non zero integer `N` value for the `deadlockaggressive` Message Store option indicates aggressive deadlock resolution, combined by delaying transaction retries by `N` seconds.

16.9 diskusagethreshold Option

Specifies the disk-usage threshold for the partition-monitoring feature. (For details about this feature, see the [checkdiskusage](#) option in Unified Configuration, or the `local.store.checkdiskusage` parameter in legacy configuration). The value of `diskusagethreshold` is a percentage from 1 to 99.

16.10 ensureownerrights Option

By default, the Message Store grants list and administer rights to a folder's owner. If the `ensureownerrights` Message Store option is set to "0", however, then these owner rights can be removed in order to create hidden folders.

16.11 expungesynclevel Option

The `expungesynclevel` Message Store option specifies the synchronization level for the store expunge file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `expungesynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `expungesynclevel`.

16.12 finalcheckpoint Option

If the `finalcheckpoint` Message Store option is set to 1, then the Message Store performs a final checkpoint of the transaction log before closing the mailbox list database.

16.13 indexsynclevel Option

The `indexsynclevel` Message Store option specifies the synchronization level for store index file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `indexsynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `indexsynclevel`.

16.14 listimplicit Option

When the `listimplicit` Message Store option is set to "1", implicitly shared folders will appear in lists performed by store admins.

16.15 logexpungedetails Option

The `logexpungedetails` Message Store option controls whether details of expunge operation will be logged. If set to "1", expunge details will be logged.

16.16 maxfolders Option

The `maxfolders` Message Store option specifies a maximum number of folders. Set to 0 (the default) for infinite.

16.17 maxlog Option

The `maxlog` Message Store option specifies the maximum number of allowable accumulated database transaction log files before the server is deemed unhealthy, after which `msprobe` will trigger a restart of `stored`.

16.18 maxmessages Option

The `maxmessages` Message Store option specifies a maximum number of messages per folder.

16.19 messagesynclevel Option

The `messagesynclevel` Message Store option specifies the synchronization level for store message file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `messagesynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `messagesynclevel`.

16.20 overquotastatus Option

The `overquotastatus` Message Store option enables tracking of user quota status, thus enabling quota enforcement before messages are enqueued in the MTA and thereby reducing the potential for MTA queues to fill 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 `mailUserStatus` LDAP attribute is set (by the Message Store) to `overquota` so no more messages will be accepted by the MTA.

16.21 perusersynclevel Option

The `perusersynclevel` Message Store option specifies the synchronization level for the store peruser file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `perusersynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `perusersynclevel`.

16.22 pin Option

The `pin` Message Store option specifies a list of IMAP mailbox names -- in common terminology, folder names -- to protect from deletion or modification except by the Message Store Administrator. The format is as follows: "`mailbox1%mailbox2%mailbox 3`", where `mailbox1`, `mailbox2` and `mailbox 3` are the IMAP mailboxes to be protected (note that spaces can be used in mailbox names), and `%` is the separator between each mailbox.

For instance, if a Messaging Server administrator has used

```
# mboxutil -r mboxname mboxname partition
```

to move certain folders to a specified partition and wishes to ensure that the folders *stay* on that specified partition, rather than getting deleted and recreated on another partition, (for instance, has moved "Trash" and "Spam" folders to less expensive storage), then the administrator will also want to "pin" the folders in question, *e.g.*:

```
msconfig> set store.pin Trash%Spam
```

16.23 quotaoverdraft Option

The `quotaoverdraft` Message Store option is used to provide compatibility with systems that migrated from the Netscape Messaging Server. When set to 1, the Message Store allows delivery of one additional message that puts disk usage over quota. After the user is over quota, any additional messages are deferred or bounced, the quota warning message is sent, and the quota grace period timer starts. The option is treated as being enabled if the `overquotastatus` option is set.

16.24 rollingdbbackup Option

The `rollingdbbackup` Message Store option controls whether rolling store database backups are made. The default is 1, meaning that such database backups are made.

16.25 seenckpinterval Option

The `seenckpinterval` Message Store option sets the peruser db archive interval (in number of hours). The default is 6 Set to 0 to disable peruser archiving.

16.26 seenckpstart Option

The `seenckpstart` Message Store option sets the initial hour of the peruser db archive after stored starts running. Allowed values are 0 (midnight) - 23 (11PM). The default is 0.

16.27 sharedfolders Option

The `sharedfolders` Message Store option enables shared folder listing and namespaces. SELECT and LSUB are not affected by this option. Users can SELECT their shared folders and use LSUB to list subscribed shared folders when this option is disabled.

16.28 snapshotdirs Option

The `snapshotdirs` Message Store option specifies the number of separate snapshots to store on disk. Minimum is 2. Recommend enough to be sure you have a good database back by the time you figure out the current one is beyond repair.

16.29 snapshotpath Option

The `snapshotpath` Message Store option specifies the path in which to copy the `mboxlist` directory. Permissions must be set for the message store owner. Snapshots will be placed in subdirectories.

16.30 subscribesynclevel Option

The `subscribesynclevel` Message Store option controls the synchronization level for store subscribe file, overriding the general `synclevel` value. 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

If `synclevel` is at its default value (-1), then `subscribesynclevel` defaults to 1. However, if `synclevel` has been set to a non-default value, then that value also becomes the default for `subscribesynclevel`.

16.31 synclevel Option

The `synclevel` Message Store option specifies the default sync level for store files. -1: no default, 0: no sync, 1: data sync only, 2: data sync and metadata sync (that is, all file attributes, including access time and modification time).

The synchronization level may be set more specifically for particular types of Message Store files, overriding this general `synclevel` setting, via the more specific options `messagesynclevel`, `cachesynclevel`, `indexsynclevel`, `expungesynclevel`, `perusersynclevel`, and `subscribesynclevel`.

16.32 backupexclude Option

The `backupexclude` Message Store option specifies mailboxes to be excluded from a backup operation. You can specify a single mailbox or a list of mailboxes separated by the "%" character.

16.33 admins Option

The `admins` Message Store option takes a space separated list of user ids with message store administrator privileges. If single-valued or not set, the MMP will use this as the default value for its `storeadmin` option.

16.34 indexeradmins Option

The `indexeradmins` Message Store option takes a space separated list of user ids with Message Store `indexer` administrator privileges. The `last access timestamp` will not be updated when authenticating with a user id in this list.

16.35 autorepair Option

The `autorepair` Message Store option may be set to repair damaged mailboxes automatically.

16.36 autorepairdebug Option

The `autorepairdebug` Message Store option enables the backup of mailbox index files to the `base.tmpdir/storeddebug` directory (`local.tmpdir/storeddebug` in legacy configuration) before repair. The maximum number of backup mailboxes is 10. Remove the files and directories under `storeddebug` manually when they are not needed.

16.37 cleanupage Option

The `cleanupage` Message Store option specifies the age (in hours) of expired or expunged message before purge will permanently remove it. The minimum allowed value is 1; the maximum is 504 (3 weeks).

16.38 cleanupsize Option

The `cleanupsize` Message Store option specifies the minimum number of expunged messages before purge will permanently remove them.

16.39 dbcachesize Option Under store

The `dbcachesize` Message Store option specifies the mailbox list database cache size. Setting the optimal cache size can make a big difference in overall Message Store performance. Cache efficiency can be determined by running `msg-svr-base/imcheck -s`.

As of Messaging Server 7.0.5, the default for the Message Store option `dbcachesize` is 67108864 (previously the default had been 16777216). Only values in the range 1048576-1073741824 (from $1024*1024$ to $1024*1024*1024$) will be used, with smaller values being silently adjusted up, while attempting to set a larger value will result in only this maximum being used (and a warning message, if warning level logging is enable for the Message Store).

Note that there is also a separate `dbcachesize` option available at the Message Store `msghash` level, `store.msghash.dbcachesize`.

16.40 dblogregionmax Option

The `dblogregionmax` Message Store option sets the size of the underlying logging area of the `mboxlist` environment, in bytes. The log region is used to store filenames and commit records. The log region size should be at least $(600 * \text{number of message store processes}) + (100 * \text{max commit records}) + 32000$. Values in the range 131072-2097152 (from $128*1024$ to $2*1024*1024$) are permitted; the default is 655360 ($640*1024$).

16.41 dbregionmax Option

The `dbregionmax` Message Store option sets the maximum amount of memory to be used by shared structures in the `mboxlist` environment region. These are the structures used to coordinate access to the environment other than mutexes and those in the page cache. Values in the range 33554432-536870912 (from $32*1024*1024$ to $512*1024*1024$) are allowed, with the default being 33554432.

16.42 folderlockcount Option

The `folderlockcount` Message Store option sets the maximum number of folder locks. The minimum allowed value is 1000; the maximum is 100000.

16.43 dbtmpdir Option

The `dbtmpdir` Message Store option specifies the mailbox list database temporary directory. Defaults to `/tmp/.ENCODED_SERVERROOT/store/`, where `ENCODED_SERVERROOT` is composed of mail server user plus the `$SERVERROOT` with `/` replaced by `_`. *e.g.* `/tmp/.mailsrv_opt_sun_comms_messaging64/store/`

This is a directory which is very heavily accessed. If the disks that house the mailbox list database temporary directory are not fast enough at very large sites, performance problems might occur. As part of their performance and tuning steps, sites should take a note of this and define a value for this parameter which either points to a memory mapped file system, or which points to a location on a fast file system.

16.44 defaultmailboxquota Option

The `defaultmailboxquota` Message Store option specifies the default mailbox quota in bytes, kilobytes, megabytes, or gigabytes, *i.e.*, 3221225472, or 3145728K, or 3072M, or 3G.

Note that the maximum value that will be handled properly is 4294967292K.

Note that there is a user level LDAP attribute for setting per-user mailbox quota (overriding for that user this Message Store `defaultmailboxquota` default quota). The user level LDAP attribute is normally named `mailQuota`, but see the [ldap_disk_quota](#) MTA option; and see also the [ldap_domain_attr_disk_quota](#) MTA option for defining a domain level LDAP attribute.

16.45 defaultmessagequota Option

The `defaultmessagequota` Message Store option specifies the default message quota (in number of messages).

Note that the maximum value that will be handled properly is 4294967292.

Note that there is a user level LDAP attribute for setting per-user message quota (overriding for that user this Message Store `defaultmessagequota` default quota). The user level LDAP attribute is normally named `mailMsgQuota`, but see the [ldap_message_quota](#) MTA option; and see also the [ldap_domain_attr_message_quota](#) MTA option for defining a domain level LDAP attribute.

16.46 defaultpartition Option

The `defaultpartition` Message Store option specifies the default partition. Only applicable on INBOX. Subfolders will be created in the partition of the parent folder.

16.47 maxcachefilesize Option

The `maxcachefilesizesize` Message Store option specifies the maximum cache file size (in bytes). A new cache file is created when the current cache file size has exceeded this limit. Minimum value is 1048576.

16.48 cachepreviewlen Option

The `cachepreviewlen` Message Store option specifies the message preview cache record length. Save the first chunk of the message body in the cache file for fast preview access.

16.49 mailboxpurgedelay Option

The `mailboxpurgedelay` Message Store option When a mailbox is deleted by the end user, expunge all the messages and purge the data after [store.cleanupage](#) has expired. Expunged messages can be restored with `'mboxutil -R'`. Expunged messages are moved to the new location when a mailbox is renamed.

16.50 encryptnew Option

RESTRICTED: The `encryptnew` Message Store option enables encrypting new messages.

16.51 keypass Option

RESTRICTED: Keystore password.

16.52 keylabel Option

RESTRICTED: The `keylabel` Message Store option specifies the label of the Message Store key in the keystore.

16.53 quotaenforcement Option

The `quotaenforcement` Message Store option enables quota enforcement. When off, the quota database is still updated, but messages are always delivered.

16.54 quotaexceededmsginterval Option

The `quotaexceededmsginterval` Message Store option specifies the interval (in days) to wait before sending another [quota exceeded message](#).

16.55 quotagraceperiod Option

The `quotagraceperiod` Message Store option specifies the time (in hours) a mailbox must be over quota before messages to the mailbox will bounce back to the sender.

16.56 quotanotification Option

The `quotanotification` Message Store option enables quota notification for the Message Store. Such notification may include both an actual e-mail message, generated at a frequency controlled by the `quotaexceededmsginterval` Message Store option, and (if the user's e-mail client supports IMAP ALERT) also a notification pop-up on the user's e-mail client screen every time the user selects a mailbox.

16.57 quotawarn Option

The `quotawarn` Message Store option specifies the percentage of quota that must be exceeded before clients are sent an over quota warning.

16.58 serviceadmingroupdn Option

The `serviceadmingroupdn` Message Store option specifies the DN of the service administrator group.

Normally initial configuration sets this option to a value of the form:

```
cn=Service Administrators,ou=Groups, <local.ugldapbasedn>
```

16.59 umask Option

The `umask` Message Store option specifies the file mode creation mask (in octal) for many files created by Messaging Server, including the MTA as well as the Message Store. By default, store file access by other users or users in the same group is forbidden by file permissions. Setting `umask` to '027' will allow users in the same primary group as the Messaging Server user to read subsequently created message store files. A more restrictive '037' setting may be appropriate for log file access. See also the `filemode` logfile option.

16.60 expiresieve Option

The `expiresieve` Message Store option enables use of [Sieve scripts](#) in expire rules.

16.61 quotaexceededmsg Option Under store

The `quotaexceededmsg` Message Store option specifies the warning message to be sent to a user when a user's quota exceeds `quotawarn` (legacy configuration, `store.quotawarn`). The message must contain a header (with at least a subject line), followed by \$\$, then the message body. The \$ represents a new line. 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.

16.62 Message Store archive options

There are a number of options relating to the Message Store's facility for performing either or both of compliance archiving and/or operational archiving. (Note that if configured to perform archiving, the Message Store generates archive message copies when users perform IMAP APPEND operations. See also [Archiving messages](#) for discussion of configuring the

MTA to generate archive message copies while messages are transitting the MTA.) Note that a list of such options and related topics may be obtained using the command:

```
msconfig> apropos archive
```

A Message Store archive option may be set using a command such as

```
msconfig> set store.archive.option-name option-value
```

See also the [store.msghash.enable](#) and [store.msghash.nummsgs](#) options.

See also the general discussion of [Archiving messages](#).

New in MS 8.0.1, the Message Store supports generating archive message copies in Microsoft Exchange "envelope journaling" format, directed to some archival address; see the [style](#), [destination](#), and [source_channel](#) Archive options in particular.

16.62.1 tmpdir Option Under archive

The `tmpdir` Message Store archive option specifies the temporary directory for archived message retrieval. If not set, the `tmpdir` Base option value (`local.tmpdir` in legacy configuration) is used.

On Linux, this option should instead be set to `/dev/shm/`.

16.62.2 compliance Option

The `compliance` Message Store archive option enables compliance archive.

16.62.3 operational Option

The `operational` Message Store archive option enables operational archive.

16.62.4 source_channel Option

The `store.archive.source_channel` option specifies the name of the [channel](#) used to submit [Microsoft Exchange Journal format messages](#). This option must be set (to the name of a valid MTA channel) when Message Store archive journal format is set (`store.archive.style=3`). It is recommended that a distinct MTA channel be created for this Message Store journal archive purpose, so that such submissions are clearly identifiable in the [MTA message transaction logs](#).

16.62.5 destination Option

If set, the `store.archive.destination` option specifies the address where [Exchange Journal format archive messages](#) generated by message store compliance archiving are to be sent. If the option is not set, archive messages are sent to the addresses specified in the [domain capture](#) and [user capture](#) LDAP attributes associated with the user performing the IMAP APPEND operation.

16.62.6 style Option

The `store.archive.style` option controls the archive format that is created by store compliance archiving. Currently the supported values are:

1. AXS:One
2. it.com format
3. Microsoft Exchange Journal format

The default value is 1, meaning AXS:One archiving is the default.

16.62.7 reportdir Option

Archive confirm report directory.

16.62.8 intext Option

The `intext` Message Store archive option controls whether or not address reversal processing is done while archiving to determine whether addresses are internal or external.

16.62.9 posteddatemode Option

Controls how the `Axs:One` `PostedDate` field is populated. A setting of 0 uses the header `Date:` field. A positive value `N` that's less than 100 uses the date information from the `N`th header `Received:` field counting from the top of the header. A negative value `N` greater than -100 uses the date information from the `(-N)`th header `Date:` field counting from the bottom of the header. A value of 100, the default, uses the IMAP message internal date.

16.62.10 useheaderrecipients Option

The `useheaderrecipients` Message Store archive option controls whether header recipient addresses (`To:`, `Cc:`, and `Bcc:` fields) are treated as specifying actual message recipients. In operational mode no envelope information is available, so header information is the only source of possible message recipients.

16.62.11 retrieveport Option

The `retrieveport` Message Store archive option specifies the archive retrieve server port.

16.62.12 retrieveserver Option

Archive retrieve server.

16.62.13 retrievetimeout Option

The `retrievetimeout` Message Store archive option specifies the archive retrieve timeout in seconds.

16.62.14 path Option Under archive

The `path` Message Store archive option specifies the archive injection directory.

16.63 Message Store checkpoint options

There are a few options relating to the Message Store's facility for checkpointing.

16.63.1 `stresslimit` Option

The `stresspoint` Message Store checkpoint option specifies the maximum stored checkpoint duration in seconds. Throttling starts when checkpoint duration exceeds this limit.

16.63.2 `debug` Option Under `checkpoint`

The `debug` Message Store checkpoint option (*i.e.*, `store.checkpoint.debug`), if set, enables stored checkpoint debug.

16.64 Message Store `dbreplicate` options

There are a number of options relating to the Message Store's facility for performing replication of its database. Note that a list of such options and related topics may be obtained using the command:

```
msconfig> apropos dbreplicate
```

A Message Store `dbreplicate` option may be set using a command such as

```
msconfig> set store.dbreplicate.option-name option-value
```

16.64.1 `enable` Option Under `dbreplicate`

The `enable` Message Store `dbreplicate` option enables the `mboxlist` database replication feature.

16.64.2 `port` Option Under `dbreplicate`

The `port` Message Store `dbreplicate` option specifies the mailbox list database replication TCP port number. This will be used to listen for incoming connections.

16.64.3 `dbremotehost` Option

The `dbremotehost` Message Store `dbreplicate` option specifies a space separated list of remote hosts in the replication group. Host format is `host[:port]`. If port is not specified, default port number 55000 is used. If this is not set, but a `storehostlist` value is set for a proxy group including the current server, then this will default to the value of that `storehostlist` setting.

16.64.4 `dbpriority` Option

The `dbpriority` Message Store `dbreplicate` options specifies the replication site priority in group elections. A special value of 0 indicates that this site cannot be a replication group master.

16.64.5 twosites Option

This option enables two sites replication. When this option is disabled, the message store cannot take over as master if the original master fails in a replication group with only two sites. In the event this happens, the message store cluster will be unavailable for write access. A two-site replication group is vulnerable to duplicate masters when there is a disruption to communications between the sites. You should strongly consider having three or more electable sites in your replication group. All sites in the group must have the same value for this option.

16.64.6 queuemax Option

The `queuemax` Message Store `dbreplicate` options specifies the replication manager incoming queue size limit.

16.64.7 ackpolicy Option

The `ackpolicy` Message Store `dbreplicate` option controls the replication manager transaction commit acknowledgment policy. All nodes in a cluster must have the same policy. Valid values are:

- 0 = Do not wait for any client replication message acknowledgments.
- 1 = Wait until at least one client site has acknowledged.
- 2 = Wait until at least one electable peer has acknowledged.
- 3 = Wait until it has received acknowledgements from the majority of electable peers.
- 4 = Wait until all electable peers have acknowledged.
- 5 = Wait until all connected clients have acknowledged.
- 6 = Wait until all replicaton clients have acknowledged.

16.64.8 acktimeout Option

The `acktimeout` Message Store `dbreplicate` options specifies the amount of time, in seconds, the replication manager waits to collect enough acknowledgments from replication group clients, before giving up and returning a failure indication.

16.65 Message Store deadlock options

Under the `deadlock` group are a few Message Store options relating to deadlocks. See also Message Store option (directly under `store`, rather than under `store.deadlock`) [deadlockaggressive](#)".

16.65.1 autodetect Option

The `autodetect` Message Store `deadlock` option sets whether all or just one thread resolves deadlock.

16.65.2 checkinterval Option

The `checkinterval` Message Store `deadlock` option specifies the sleep length in milliseconds between deadlock detections. Note that previous versions of the documentation

for this option (such as provided by `configutil -H`) incorrectly stated the units as microseconds, but the implementation has always interpreted this as milliseconds.

16.66 Message Store expire options

`exploglevel` is presently the only Message Store option specifically set under the `expire` group.

Several former `expire` Message Store options have been [deleted](#).

Additional Unified Configuration Message Store options relating to message expiry (but which are not specifically `store.expire` options) include:

- `store.expirestart`
- `store.expiresieve`
- `store.expirerule.* options`
- `store.cleanupage`
- `store.mailboxpurgedelay`

Also see the Scheduler options:

- `schedule.task:expire.enable`
- `schedule.task:expire.crontab`

16.66.1 exploglevel Option

The `exploglevel` Message Store `expire` option specifies an expire log level: 0: no log. 1: log summary for the entire expire session. 2: log one message per mailbox expired. 3: log one message per message expired.

16.66.2 crontab Option Use With expire Under task

The `crontab` Scheduler `task` option for the `expire` task controls the interval for running `imexpire`, enabled with `schedule.task:expire.enable` (Unified Configuration) which defaults to the setting of the `store.enable` setting, (or in legacy configuration, `local.schedule.expire.enable` which defaults to the setting of `local.store.enable`). `schedule.task:expire.crontab` uses UNIX `crontab` format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration normally sets this to:

```
msconfig> show schedule.task:expire.crontab
role.schedule.task:expire.crontab = 0 23 * * * bin/imexpire
```

16.67 Message Store expirerule options

There are a number of options relating to the Message Store's rules for expiring (purging) old (or otherwise undesired) messages. Note that a list of such expiration rule options and related topics may be obtained using the command:

```
msconfig> apropos expirerule
```

A Message Store `expirerule` option may be set using a command such as

```
msconfig> set store.expirerule.option-name option-value
```

See also [Message Store expire options](#) for a list of additional Message Store options relating to message expiry (but which are not `expirerule` options), and see the [expiresieve](#) Message Store option which can enable use of Sieve filter tests in expire rules.

16.67.1 deleted Option

Syntax: "and"|"or". When the `deleted` Message Store `expirerule` option is set to "or", expire messages if the 'Deleted' system flag is set or other criteria in the expire rule are met. When set to "and", expire messages if the 'Deleted' system flag is set and other criteria in the expire rule are met.

16.67.2 exclusive Option

When the `exclusive` Message Store `expirerule` option is set to "1", it is the only rule applied even if other rules match the given criteria.

16.67.3 folderpattern Option

The `folderpattern` Message Store `expirerule` option specifies the folders for which the rule apply. The value must start with a "user/", which represents the directory `store-root/partition/*/. Syntax: POSIX regular expression. For examples, refer to imexpire Folder Patterns Using Regular Expressions in the Admin Guide.`

16.67.4 foldersizebytes Option

The `foldersizebytes` Message Store `expirerule` option specifies the maximum number of bytes in folder. Older messages will be expunged if this limit is exceeded.

16.67.5 messagecount Option

The `messagecount` Message Store `expirerule` option specifies the upper limit on number of messages to be kept in the specified folders. Older messages will be expunged if this limit is exceeded.

16.67.6 messagedays Option

The `messagedays` Message Store `expirerule` option specifies the upper limit on how long a message is kept in the specified folders (in days).

16.67.7 messagesize Option

The `messagesize` Message Store `expirerule` option specifies the size of an over-sized message.

16.67.8 messagesizedays Option

The `messagesizedays` Message Store `expirerule` option specifies the days an [over-sized message](#) should remain in a folder.

16.67.9 seen Option

Syntax: "and"|"or". When the `seen` Message Store `expirerule` option is set to "or", expire messages if the 'Seen' system flag is set or other criteria in the expire rule are met. When set to "and", expire messages if the 'Seen' system flag is set and other criteria in the expire rule are met.

16.68 Message Store folderquota options

Currently, the only Message Store `folderquota` option is [enable](#).

16.68.1 enable Option Under folderquota

The `enable` Message Store `folderquota` option enables quota by folder.

16.69 Message Store messagetype and typequota options

The Message Store `messagetype` option [enable](#) enables Message Store message typing; by default, the standard MIME Content-type: header field is inspected, but optionally the `messagetype` option [header](#) may be set, for sites that wish to inspect an alternate header field (the Message-Context: header field, defined in [RFC 3458 \(Message Context for Internet Mail\)](#) can be a good choice); then under `messagetype` the further integer-indexed `mtindex` groups of options [contenttype](#) and [flagname](#) specify the actual message types, and optionally a corresponding [quotaroot](#) may be set for purposes of type-specific quotas. That is, with the [header](#) Message Store `messagetype` option specifying what header field to inspect, then integer-indexed sets of `mtindex` options define the types themselves, where an integer-indexed `mtindex` `contenttype` option defines a value to recognize in that header field, and then the correspondingly indexed [flagname](#) Message Store `messagetype` `mtindex` option specifies the Message Store name to use for such messages; *e.g.*:

```
msconfig> set store.messagetype.enable 1
msconfig# set store.messagetype.header Message-context
msconfig# set store.messagetype.mtindex:1.contenttype voice-message
msconfig# set mtindex:1.flagname Voice
msconfig# set mtindex:2.contenttype fax-message
msconfig# set mtindex:2.flagname Fax
msconfig# set mtindex:3.contenttype pager-message
msconfig# set mtindex:3.flagname Pager
msconfig# set mtindex:4.contenttype multimedia-message
msconfig# set mtindex:4.flagname Multimedia
msconfig# set mtindex:5.contenttype text-message
msconfig# set mtindex:5.flagname Text
```

Sites that wish to perform Message Store message typing that find they are receiving messages lacking the header line which the site wishes to use as a basis for message typing decisions,

(*e.g.*, receiving messages lacking Message-Context:), may wish to consider configuring an MTA system level [Sieve filter](#) to add an appropriate header line based upon other message characteristics (*e.g.*, an addheader action based upon the message's outermost MIME Content-type: header line) to provide a basis for the Message Store's message typing.

Sites that wish to enforce per-message-type quotas would also set the [enable Message Store typequota option](#), and define per-message-type quotas using the [quotaroot](#) option under Message Store `messagetype` with `mtindex` appropriately indexed for each type. For instance, continuing the above example (that is, assuming that five message types have already been defined as above), enabling use of user per-message-type quotas would involve:

```
msconfig# set store.typequota.enable 1
msconfig# set store.messagetype.mtindex:1.quotaroot Voice
msconfig# set mtindex:2.quotaroot Fax
msconfig# set mtindex:3.quotaroot Pager
msconfig# set mtindex:4.quotaroot Multimedia
msconfig# set mtindex:5.quotaroot Text
```

and then setting a user's actual quotas would mean setting the user's MailQuota LDAP attribute (quota on space) and/or mailMsgQuota LDAP attribute (quota on number of messages) with per-message-type value syntax; *e.g.*:

```
mailQuota: 80M;#Voice%40M;#Fax%10M;#Pager%5M;#Multimedia%20M;#Text%10M
mailMsgQuota: 10000;#Voice%300;#Fax%40;#Pager%300;#Multimedia%300;#Text%9000
```

16.69.1 enable Option Under messagetype

The `enable Message Store messagetype` option enables the Message Store's message typing feature.

16.69.2 enable Option Under typequota

The `enable Message Store typequota` option is checked only if Message Store message typing has been enabled (`store.messagetype.enable` is 1). When Message Store message typing has been enabled, then setting `store.typequota.enable` enables quota by message type. To set different quotas for different message types, see the [quotaroot](#) Message Store `messagetype mtindex` option.

16.69.3 header Option

By default, Message Store message typing (if enabled -- see the [store.messagetype.enable](#) option) determines message type from the MIME Content-type: header field. The `header Message Store messagetype` option allows a site to inspect an alternate header field for Message Store message typing purposes.

16.69.4 contenttype Option

The `contenttype Message Store messagetype` option defines a message type. More specifically, each integer-indexed `mtindexcontenttype` value defines a string to recognize on the [store.messagetype.header](#) field, whereas the correspondingly integer-

indexed [flagname](#) Message Store `message_type mtindex` option specifies the Message Store name to use for such messages.

16.69.5 flagname Option

The `flagname` Message Store `message_type mtindex` option specifies the flag name of a message type. The `flagname` option would always be used in conjunction with a (correspondingly indexed `mtindex`) [contenttype](#) option.

16.69.6 quotaroot Option

The `quotaroot` Message Store `message_type mtindex` option specifies the quota root suffix for a message type. (See the IMAP QUOTA extension, [RFC 2087](#).) The `quotaroot` option would always be used in conjunction with a (correspondingly indexed `mtindex`) [contenttype](#) and [flagname](#) options.

Note that enforcement of different quotas for different message types will not occur unless the [enable](#) Message Store `typequota` option has been set (and of course also the user's LDAP entry must define different quotas in its `mailQuota` and/or `mailMsgQuota` LDAP attributes).

16.70 Message Store msghash options

A Message Store `msghash` option may be set using a command such as

```
msconfig> set store.msghash.option-name option-value
```

16.70.1 enable Option Under msghash

The `enable` Message Store `msghash` option enables message hash database.

16.70.2 dbcachesize Option Under msghash

The `dbcachesize` Message Store `msghash` option, `store.msghash.dbcachesize`, specifies the message hash database cache size. Only values in the range 524288-536870912 (from 512×1024 to $512 \times 1024 \times 1024$) will be used (other values will be silently adjusted into that range), with the default being 8388608 ($8 \times 1024 \times 1024$).

Note that this `store.msghash.dbcachesize` option is separate from the [store.dbcachesize](#) option, which controls a different cache size.

16.70.3 nummsgs Option

The `nummsgs` Message Store `msghash` option specifies the message hash database size.

16.71 Message Store purge options

The Message Store has several options under the `purge` group:

- [count](#)
- [enable](#)

- [maxthreads](#)
- [percentage](#)

A Message Store purge option may be set using a command such as

```
msconfig> set store.purge.option-name option-value
```

Additional Message Store options that relate to purging of messages but which are *not* under purge include:

- [store.cleanupage](#)
- [store.cleanupsize](#)
- [store.mailboxpurgedelay](#)
- [store.relinker.maxage](#)
- [store.relinker.purgecycle](#)

Also see the [Scheduler task options](#)

- [schedule.task:expire.enable](#)
- [schedule.task:expire.crontab](#)

16.71.1 enable Option Under purge

The enable Message Store purge option, `store.purge.enable` (Unified Configuration) or `local.purge.enable` (legacy configuration), enables the purge server on `start-msg` startup. Initial configuration normally sets this option based on the setting of the [enable](#) Message Store option, `store.enable` (Unified Configuration) or `local.store.enable` (legacy configuration).

Note that a setting of `store.purge.enable=1` (whether set explicitly, or in effect due to `store.enable=1`) enables the Message Store `impurge` process to run as a daemon (server) process. Alternatively, it is possible to instead disable such a daemon process, `store.purge.enable=0`, and instead configure the [Scheduler](#) to execute an `impurge` command periodically. `impurge` cannot be executed manually by an administrator, not run by the Scheduler, if it is already running as a daemon process; attempting to run it again will result in an error in the Messaging Server `default` log file along the lines of:

```
24/Feb/2009:14:47:15 +1100] hostname impurge[17986]: General Error: Could not get purge session lock. Possibly another impurge is running
```

16.71.2 count Option

The count Message Store purge option specifies the minimum number of expunged cache records (for a folder) before purge will permanently remove them. The minimum allowed value is 1; maximum is 1000000.

16.71.3 maxthreads Option Under purge

The `maxthreads` Message Store purge option specifies the maximum number of threads. The default is 64; the maximum value that will be used is 100; attempts to set a higher value will result in a value of 100 getting used.

16.71.4 percentage Option

The percentage Message Store purge option specifies the percentage of expunged cache records (for a folder) before purge will permanently remove them.

16.71.5 crontab Option Use With purge Under task

The crontab Scheduler task option for the purge task controls the interval for running `imsimta purge`, enabled with `schedule.task:purge.enable` (Unified Configuration) which defaults to the value of `mta.enable`, (or in legacy configuration `local.schedule.purge.enable` which defaults to the value of `local.imta.enable`). `imsimta purge` removes older MTA log files. `schedule.task:purge.crontab` uses UNIX crontab format: minute hour day-of-month month-of-year day-of-week command arguments.

Initial configuration sets this to:

```
msconfig> show schedule.task:purge.crontab
role.schedule.task:purge.crontab = 0 0,4,8,12,16,20 * * * bin/imsimta purge -num=5
```

16.72 Message Store relinker options

A Message Store relinker option may be set using a command such as

```
msconfig> set store.relinker.option-name option-value
```

16.72.1 enable Option Under relinker

The `enable` Message Store relinker option enables real-time re-linking of messages in the append code, and `stored purge`. The `relinker` command-line tool may be run even if this option is off, however since `stored` will not purge the repository, `relinker -d` must be used for this task. Turning this option on affects message delivery performance in exchange for the disk space savings.

16.72.2 maxage Option

The `maxage` Message Store relinker option specifies the maximum age in hours for messages to be kept in the repository, or considered by the `relinker` command-line. `-1` means no age limit, that is, only purge orphaned messages from the repository. For `relinker` it means process existing messages regardless of age. Shorter values keep the repository smaller thus allow `relinker` or `stored purge` to run faster and reclaim disk space faster, while longer values allow duplicate message re-linking 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.

16.72.3 minsize Option

The `minsize` Message Store relinker option specifies the minimum size in kilobytes for messages to be considered by run-time or command-line `relinker`. Setting a non-zero value gives up the `relinker` benefits for smaller messages in exchange for a smaller repository.

16.72.4 purgecycle Option

The `purgecycle` Message Store `relinker` option specifies the approximate duration in hours of an entire stored 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 stored (then purge must be performed using the `relinker -d` command).

16.73 Message Store shared folder options

Message Store `privatesharedfolders` options include:

- `restrictanyone`
- `restrictdomain`
- `shareflags`

Message Store `publicsharedfolders` options include:

- `user`

Other Message Store options relating to shared folders include:

- `store.listimplicit`
- `store.sharedfolders`

Other components with options relating to shared folders include:

- `proxyserverlist` base option
- `imap.immediateflagupdate`

16.73.1 restrictanyone Option

The `restrictanyone` Message Store `privatesharedfolders` option disallows regular users sharing private folders to anyone.

16.73.2 restrictdomain Option

The `restrictdomain` Message Store `privatesharedfolders` option disallow regular users sharing private folders to users in another domain.

16.73.3 shareflags Option

The `shareflags` Message Store `privatesharedfolders` option controls whether private shared folders share `\Seen` and `\Deleted` flags across users.

16.73.4 user Option Under publicsharedfolders

The `user` Message Store `publicsharedfolders` option specifies the public shared folder owner's store user identity.

16.74 Message Store deleted options

A number of Message Store options have been deleted.

16.74.1 backupbadidxfiles Option

Enable backups of bad index files. DELETED: Feature never implemented.

16.74.2 listrecover Option

Specifies how LIST command is done in respects to recovery. DELETED: not fully implemented.

16.74.3 local.store.maxlogs Option

Specifies a maximum number of allowable accumulated log files. DELETED: Instead use the [maxlog](#) Message Store option in Unified Configuration, or the `local.store.maxlog` parameter in legacy configuration.

16.74.4 messagetypeplugin Option

Full pathname and command line arguments (preceded by a "\$" character) for message typing plugin. DELETED: Appears code to actually load plug-ins was not present in 6.3 or later.

16.74.5 local.store.serversidewastebasket Option

Enables server side wastebasket. DELETED: Never used.

16.74.6 snapshotinterval Option

Interval of time between snapshots in minutes. It is recommended that you perform this procedure at least once a day. DELETED: Removed in Messaging Server 7.0

16.74.7 defaultacl Option

Default ACL. DELETED: Removed in Messaging Server 6.3.

16.74.8 diskflushinterval Option

DELETED: Use a `*synclevel` option (Unified Configuration) or `local.store.*synclevel` parameter (legacy configuration) instead.

16.74.9 expirestart Option

`imexpire` start time. Format: 0-23 (represents hour). DELETED: Use `local.schedule.expire` (legacy configuration) or `schedule.task:expire.crontab` (Unified Configuration) instead.

16.74.10 mailboxexpungesize Option

Size (in bytes) of expired or expunged message before purge will permanently remove it. Minimum value is 1048576. DELETED: Use [store.cleanupsize](#) instead.

16.74.11 cleanonly Option

Perform purge only, do not perform `imexpire`. DELETED: (removed as of Messaging Server 7.0)

16.74.12 workday Option

Perform expire/cleanup on this day of the week. Values: 0-6 (0= Sunday). DELETED: Instead use `local.schedule.expire` (legacy configuration) or `schedule.task:expire.crontab` (Unified Configuration).



Chapter 17 message_language options

17.1 quotaexceededmsg Option Under message_language	17-1
17.2 welcomemsg Option Under message_language	17-1

There are a few options available for localizing certain automatically generated messages.

17.1 quotaexceededmsg Option Under message_language

The `quotaexceededmsg` option under a named `message_language` group specifies a localized message to be sent to user when quota exceeds `store.quotawarn`. The message must contain a header (with at least a subject line), followed by `$$`, then the message body. The `$` represents a new line. 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.

17.2 welcomemsg Option Under message_language

The `welcomemsg` option specified under `message_language` specifies a localized welcome message for new Message Store users. The maximum size is 1 MB. Syntax: "\$" line separators, with headers.



Chapter 18 Partition options

18.1 messagepath Option	18-1
18.2 cachepath Option	18-1
18.3 path Option Under partition	18-1
18.4 path Option Use With primary Under partition	18-1

There are several options specifically concerning Message Store partitions. A partition option may be set using a command such as:

```
msconfig> set partition:primary.path partition1
```

For other, general Message Store configuration, see the [Message Store options](#).

18.1 messagepath Option

The messagepath partition option specifies the path name of a store partition containing the message files. If this is not a full path, this is relative to the store/partition subdirectory of the data directory. If not specified, the value for the [partition:partition-name.path](#) option (in legacy configuration, the store.partition.*.path configutil parameter) is used.

18.2 cachepath Option

The cachepath partition option specifies the path name of a store partition containing the mailbox cache files. If this is not a full path, this is relative to the store/partition subdirectory of the data directory. If not specified, the value for the [partition:partition-name.path](#) option (in legacy configuration, the store.partition.*.path configutil parameter) is used.

18.3 path Option Under partition

The path partition option specifies the path name of a store partition. If this is not a full path, this is relative to the store/partition subdirectory of the data directory.

The partition:partition-name.path value is also used as the default for the [messagepath](#) and [cachepath](#) partition options, if they are not set explicitly.

18.4 path Option Use With primary Under partition

The partition:primary.path option specifies the path name of the primary partition; (if not a full path, this is relative to the store/partition subdirectory of the data directory).



Chapter 19 backup_group options

19.1 re_pattern Option	19-1
------------------------------	------

Under backup_group, there is a single option, [re_pattern](#).

19.1 re_pattern Option

The `re_pattern` option under `backup_group` specifies the regular expression pattern used to select a group of user mailboxes for backup. If no patterns are specified for any backup groups, the backup group 'user' can be used (defined with the pattern '%') which matches all users.



Chapter 20 Client access to Message Store servers

The TCP client access control mechanism used by [Message Store](#) servers such as the POP and IMAP servers, and proxy servers such as the MMP and MSHTTP, uses [TCP wrappers](#). The ENS server also uses this mechanism.

Note that the MMP behaves a bit differently with respect to access control than do the other services, in that the MMP "imap" service controls both IMAP and IMAP+SSL services; that is, it controls both ports 143 and 993. In contrast, other Messaging Server services treat IMAP and IMAP+SSL as separate services, each with their own separate access control.

For access control on the MTA's SMTP server and other [Dispatcher](#) services, see instead [Mail filtering and access control](#) and in particular the [PORT_ACCESS mapping table](#).



Chapter 21 IMAP options

21.1 enable Option Under imap	21-3
21.2 port Option Under imap	21-3
21.3 enablesslport Option Under imap	21-3
21.4 legacy_proxyauth Option	21-3
21.5 adminbypassquota Option	21-3
21.6 fixinternaldate Option	21-3
21.7 immediateflagupdate Option	21-3
21.8 logcommands Option	21-4
21.9 maxnoops Option	21-4
21.10 polldelay Option	21-4
21.11 maxsearchmailboxes Option	21-4
21.12 maxmessagesize Option Under imap	21-4
21.13 submituser Option	21-4
21.14 withinresolution Option	21-5
21.15 capability_acl Option	21-5
21.16 capability_annotate Option	21-5
21.17 capability_binary Option	21-5
21.18 capability_catenate Option	21-5
21.19 capability_children Option	21-5
21.20 capability_condstore Option	21-5
21.21 capability_context_search Option	21-5
21.22 capability_context_sort Option	21-5
21.23 capability_enable Option	21-5
21.24 capability_earch Option	21-6
21.25 capability_esort Option	21-6
21.26 capability_id Option	21-6
21.27 capability_idle Option	21-6
21.28 capability_imap4 Option	21-6
21.29 capability_imap4rev1 Option	21-6
21.30 capability_language Option	21-6
21.31 capability_list_status Option	21-6
21.32 capability_literal Option	21-6
21.33 capability_login_referrals Option	21-7
21.34 capability_metadata Option	21-7
21.35 extra_capabilities Option	21-7
21.36 capability_multisearch Option	21-7
21.37 capability_namespace Option	21-7
21.38 capability_notify Option	21-7
21.39 capability_qresync Option	21-7
21.40 capability_quota Option	21-8
21.41 capability_sasl_ir Option	21-8
21.42 capability_searchres Option	21-8
21.43 capability_sort Option	21-8
21.44 capability_sort_display Option	21-8
21.45 capability_special_use Option	21-8
21.46 capability_create_special_use Option	21-8
21.47 capability_starttls Option Under imap	21-8
21.48 capability_thread_references Option	21-9
21.49 capability_thread_subject Option	21-9
21.50 capability_uidplus Option	21-9

21.51	<code>capability_unselect</code> Option	21-9
21.52	<code>capability_urlauth</code> Option	21-9
21.53	<code>capability_within</code> Option	21-9
21.54	<code>capability_x_netscape</code> Option	21-9
21.55	<code>capability_x_orcl_as</code> Option	21-9
21.56	<code>capability_x_sun_imap</code> Option	21-9
21.57	<code>capability_x_sun_sort</code> Option	21-10
21.58	<code>capability_x_unauthenticate</code> Option	21-10
21.59	<code>capability_xrefresh</code> Option	21-10
21.60	<code>capability_xsender</code> Option	21-10
21.61	<code>capability_xserverinfo</code> Option	21-10
21.62	<code>capability_xuml</code> Option	21-10
21.63	<code>allowanonymouslogin</code> Option Under <code>imap</code>	21-10
21.64	<code>banner</code> Option Under <code>imap</code>	21-10
21.65	<code>domainallowed</code> Option Under <code>imap</code>	21-10
21.66	<code>domainnotallowed</code> Option Under <code>imap</code>	21-11
21.67	<code>actions</code> Option	21-11
21.68	<code>actionattributes</code> Option	21-11
21.69	<code>enableuserlist</code> Option Under <code>imap</code>	21-11
21.70	<code>forcetelemetry</code> Option Under <code>imap</code>	21-11
21.71	<code>idletimeout</code> Option Under <code>imap</code>	21-11
21.72	<code>logprotocolerrors</code> Option Under <code>imap</code>	21-11
21.73	<code>logauthsessionid</code> Option	21-11
21.74	<code>maxprotocolerrors</code> Option Under <code>imap</code>	21-11
21.75	<code>maxsessions</code> Option Under <code>imap</code>	21-12
21.76	<code>maxthreads</code> Option Under <code>imap</code>	21-12
21.77	<code>numprocesses</code> Option Under <code>imap</code>	21-12
21.78	<code>plaintextmncipher</code> Option Under <code>imap</code>	21-12
21.79	<code>sslcachesize</code> Option Under <code>imap</code>	21-12
21.80	<code>sslnicknames</code> Option Under <code>imap</code>	21-12
21.81	<code>sslport</code> Option Under <code>imap</code>	21-12
21.82	<code>sslusessl</code> Option Under <code>imap</code>	21-13
21.83	<code>logunauthsession</code> Option Under <code>imap</code>	21-13
21.84	IMAP password expiration alert options	21-13
21.84.1	<code>firstwarn</code> Option	21-13
21.84.2	<code>viametermaid</code> Option	21-13
21.84.3	<code>metermaidtable</code> Option	21-13

There are many options affecting IMAP operation.

See also the following options which are described rather generically under [Base options](#), but which may also be set specifically under `imap` (as for instance if one wishes to have IMAP use a different value than the general base value): the various `bg*` options, and `defaultdomain`.

See also the [Indexer options](#); though they are set under `imap.indexer`, they are documented separately.

Of notable relevance to IMAP operation, see also the `base.obsoleteimap`, `base.threadholddelay`, `base.dnsresolveclient`, `base.pwchangeurl` options, and on Solaris, see also the `base.preferpoll` option.

Note that `msprobe` can probe for whether the IMAP server is running; see `msprobe`'s `msprobe.probe:imap` options.

21.1 enable Option Under imap

The `enable` IMAP option, (`imap.enable` in Unified Configuration, or `service.imap.enable` in legacy configuration), enables the IMAP service on `start-msg` startup. Note: IMAP over SSL service is enabled/disabled separately using `imap.enablesslport` in Unified Configuration, or `service.imap.enablesslport` in legacy configuration.

This option defaults to 0 if not set, but initial configuration may enable the option as appropriate.

21.2 port Option Under imap

The `port` IMAP option specifies the IMAP server port number.

21.3 enablesslport Option Under imap

The `enablesslport` IMAP option sets whether or not IMAP over SSL service is started; if enabled, this service uses the port set in the `sslport` IMAP option. For the 7.0.5 release, the `sslusessl` option must also be explicitly set to enable the separate SSL port. For the 8.0 release, setting this option enables the separate SSL port and it is no longer necessary to explicitly set the `sslusessl` option.

21.4 legacy_proxyauth Option

The `legacy_proxyauth` IMAP option enables the legacy proxy authorization IMAP `PROXYAUTH` command; this command was useful prior to 1999 when the standards-based replacement mechanism was published. This non-standard command, provides a mechanism that is similar but inferior to the standard SASL authorization identity that can be provided with the SASL PLAIN mechanism, as documented in [RFC 2595](#). This option is available for customers who have not yet adapted their systems to use the standard mechanism. Note that the `PROXYAUTH` command is not compatible with the MMP (it may or may not work).

21.5 adminbypassquota Option

Enabling the `adminbypassquota` IMAP option allows admin users to bypass [quota enforcement](#) when they append messages to mailboxes with the IMAP `APPEND` command.

21.6 fixinternaldate Option

The `fixinternaldate` IMAP option specifies whether to fix the IMAP `internaldate` for appended messages when the client fails to pass a valid date argument.

21.7 immediateflagupdate Option

When the `immediateflagupdate` IMAP option is set to 1, then Seen and Deleted flags for users other than the mailbox owner are updated in the database on disk immediately, instead

of being buffered and updated once in a while. This is needed for IMAP IDLE to show flag changes correctly with shared folders.

The default was changed to 1 for the 7.0.5 release.

21.8 logcommands Option

If the `logcommands` IMAP option is enabled, this will record information about IMAP client commands in a file called `imapcmd` in the log directory. Each line will have a session identifier followed by command information. This differs from the [telemetry facility](#) in that user-specific information is omitted to protect privacy, server responses are omitted, there is no timing information, it is controlled globally for all users and commands prior to authentication are logged. This option is refreshable (via a `refresh imapd` command) and is intended to gather a sample of client behavior rather than to be used continuously. The present version does not have the ability to size limit or rollover this log file.

21.9 maxnoops Option

The `maxnoops` IMAP option specifies the maximum number of NOOP commands accepted before connection is forcibly closed.

21.10 polldelay Option

Solaris-only. The `polldelay` (IMAP and MMP) option specifies the wait time before calling `poll()` in milliseconds. Workaround for poll performance bug on Solaris (6438988, 6379476). Setting this to -1 activates a different workaround as of 7 update 4 patch 24. The alternate code tries to keep the size of the poll array relatively constant and instead uses -1 in the poll array for inactive descriptors. The poll array will be larger, but change size less frequently. To date this appears to noticeably improve performance under stress.

The default has changed from 1 to -1 in the Messaging Server 7.0.5 release. In addition, `poll` is no longer used in the Messaging Server 7.0.5 release (and thus this option is ignored) unless [preferpoll](#) is set.

21.11 maxsearchmailboxes Option

The `maxsearchmailboxes` IMAP option specifies the maximum number of mailboxes that may be searched by one IMAP ESEARCH command. If this limit is exceeded, the search command will return an error. Setting this to 0 results in no limit.

21.12 maxmessagesize Option Under imap

The `maxmessagesize` IMAP option specifies the maximum message size (in bytes) that IMAP clients are allowed to save via the append command.

21.13 submituser Option

The `submituser` IMAP option specifies the Message Store userid used by the MTA when resolving submit IMAP URLs in [BURL](#) commands.

21.14 withinresolution Option

The `withinresolution` IMAP option specifies the interval (in minutes) between recalculations of Contexts involving the search options `YOUNGER` or `OLDER`.

21.15 capability_acl Option

The `capability_acl` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the ACL IMAP extension.

21.16 capability_annotate Option

The `capability_annotate` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the ANNOTATE-EXPERIMENT-1 IMAP extension.

21.17 capability_binary Option

The `capability_binary` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the BINARY IMAP extension.

21.18 capability_catenate Option

The `capability_catenate` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CATENATE IMAP extension.

21.19 capability_children Option

The `capability_children` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CHILDREN IMAP extension.

21.20 capability_condstore Option

The `capability_condstore` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CONDSTORE IMAP extension.

21.21 capability_context_search Option

The `capability_context_search` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CONTEXT=SEARCH IMAP extension.

21.22 capability_context_sort Option

The `capability_context_sort` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CONTEXT=SORT IMAP extension.

21.23 capability_enable Option

The `capability_enable` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the ENABLE IMAP extension.

21.24 `capability_esearch` Option

The `capability_esearch` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the ESEARCH IMAP extension.

21.25 `capability_esort` Option

The `capability_esort` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the ESORT IMAP extension.

21.26 `capability_id` Option

The `capability_id` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the IMAP ID extension.

21.27 `capability_idle` Option

Setting the `capability_idle` IMAP option to 1 (the default) causes the IMAP server to advertise and enable the IDLE IMAP extension.

21.28 `capability_imap4` Option

Advertise the IMAP4 capability. The default is normally 0 (false), but the default is 1 (true) if `obsoleteimap` (in legacy configuration, `local.obsoleteimap`) is set.

21.29 `capability_imap4rev1` Option

The `capability_imap4rev1` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the IMAP4rev1 capability.

21.30 `capability_language` Option

The `capability_language` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the LANGUAGE IMAP extension.

21.31 `capability_list_status` Option

The `capability_list_status` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the LIST-STATUS IMAP extension. This extension was initially introduced in the 8.0 release of Messaging Server and is described in [RFC 5258 \(IMAPv4 LIST Command Extensions\)](#).

21.32 `capability_literal` Option

The `capability_literal` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the LITERAL+ IMAP extension.

21.33 `capability_login_referrals` Option

The `capability_login_referrals` IMAP option, when set to 1 (the default), causes the back-end IMAP server to advertise the LOGIN-REFERRALS IMAP extension. LOGIN OK referrals are generated when the user connects to a mailstore that is not that user's primary read-write mailstore. Presently only OK referrals are generated because the client may be connecting to access shared folders owned by another user who is local to that server. The MMP does not generate login referrals but will follow them as part of the store failover feature.

21.34 `capability_metadata` Option

The `capability_metadata` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the METADATA IMAP extension.

21.35 `extra_capabilities` Option

If the `extra_capabilities` IMAP option is set, the string specified is included in the IMAP server capability list. The use of capability names that do not begin with 'X' will often break IMAP standards compliance and manifest as client compatibility problems that may result in support calls. As a result, use of this option is not recommended. Including ']' in this value will also violate the standard and break client compatibility.

21.36 `capability_multisearch` Option

For 7.0.5.31.0 and later, the `capability_multisearch` IMAP option is disabled by default and if enabled will advertise the experimental XMSEARCH IMAP extension that is a subset of the functionality described in [RFC 6237 \(IMAP4 Multimapbox SEARCH Extension\)](#). The ESEARCH command will be disabled unless this is set to 1.

For the 8.0 release and later, this is enabled by default and controls visibility of the MULTISEARCH capability as described in [RFC 6237](#).

21.37 `capability_namespace` Option

The `capability_namespace` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the NAMESPACE IMAP extension.

21.38 `capability_notify` Option

The `capability_notify` IMAP option, when set to 1, causes the IMAP server to advertise the NOTIFY IMAP extension. The default is 0.

21.39 `capability_qresync` Option

The `capability_qresync` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the QRESYNC IMAP extension.

21.40 capability_quota Option

The `capability_quota` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the QUOTA IMAP extension.

21.41 capability_sasl_ir Option

The `capability_sasl_ir` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the SASL-IR IMAP extension.

21.42 capability_searchres Option

The `capability_searchres` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the SEARCHRES IMAP extension.

21.43 capability_sort Option

The `capability_sort` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the SORT IMAP extension ([RFC 5256](#)).

21.44 capability_sort_display Option

The `capability_sort_display` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the SORT=DISPLAY IMAP extension ([RFC 5957](#)).

21.45 capability_special_use Option

The `capability_special_use` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the SPECIAL-USE IMAP extension. This was introduced in the 8.0 release of Messaging Server and is described in [RFC 6154 \(IMAP LIST Extension for Special-Use Mailboxes\)](#).

21.46 capability_create_special_use Option

The `capability_create_special_use` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the CREATE-SPECIAL-USE IMAP extension. This was introduced in the 8.0 release of Messaging Server and is described in [RFC 6154 \(IMAP LIST Extension for Special-Use Mailboxes\)](#).

21.47 capability_starttls Option Under imap

The `capability_starttls` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the STARTTLS IMAP extension.

21.48 `capability_thread_references` Option

The `capability_thread_references` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `THREAD=REFERENCES` IMAP extension.

21.49 `capability_thread_subject` Option

The `capability_thread_subject` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `THREAD=ORDEREDSUBJECT` IMAP extension.

21.50 `capability_uidplus` Option

The `capability_uidplus` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `UIDPLUS` IMAP extension.

21.51 `capability_unselect` Option

The `capability_unselect` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `UNSELECT` IMAP extension.

21.52 `capability_urlauth` Option

The `capability_urlauth` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `URLAUTH` IMAP extension.

21.53 `capability_within` Option

The `capability_within` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `WITHIN` IMAP extension.

21.54 `capability_x_netscape` Option

The `capability_x_netscape` IMAP option, if set to 1, (the default being 0), causes the IMAP server to advertise the `X-NETSCAPE` IMAP extension.

21.55 `capability_x_orcl_as` Option

The `capability_x_orcl_as` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `X-ORCL-AS` IMAP capability. This capability was added in the 8.0 release of Messaging Server and refers to protocol extensions that may be useful for ActiveSync gateways.

21.56 `capability_x_sun_imap` Option

The `capability_x_sun_imap` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the `X-SUN-IMAP` IMAP extension.

21.57 capability_x_sun_sort Option

The `capability_x_sun_sort` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the X-SUN-SORT IMAP extension.

21.58 capability_x_unauthenticate Option

The `capability_x_unauthenticate` IMAP option, when set to 1, causes the IMAP server to advertise the X-UNAUTHENTICATE IMAP extension. The default is now 0, whereas the default in Messaging Server 7.0.4 was 1.

21.59 capability_xrefresh Option

The `capability_xrefresh` IMAP option, when set to 1 (the default), causes the IMAP server to Advertise the XREFRESH IMAP extension.

21.60 capability_xsender Option

The `capability_xsender` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the XSENDER IMAP extension.

21.61 capability_xserverinfo Option

The `capability_xserverinfo` IMAP option, if set to 1 (the default being 0), causes the IMAP server to advertise the XSERVERINFO IMAP extension.

21.62 capability_xum1 Option

The `capability_xum1` IMAP option, when set to 1 (the default), causes the IMAP server to advertise the XUM1 IMAP extension.

21.63 allowanonymouslogin Option Under imap

The `allowanonymouslogin` IMAP option enables the SASL ANONYMOUS mechanism for use by IMAP.

21.64 banner Option Under imap

The `banner` IMAP option specifies the IMAP protocol welcome banner. The value is a one line string, with virtual parameters: %h=hostname, %p=protocol(ESMTP,POP or IMAP), %P=Product Name, %v and %V=Version (short or long).

21.65 domainallowed Option Under imap

The `domainallowed` IMAP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed IMAP access.

21.66 domainnotallowed Option Under imap

The `domainnotallowed` IMAP option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed IMAP access.

21.67 actions Option

The `actions` option (available for `imap`, `pop`, and `messagetrace`) specifies the actions enabled in Message Store transaction logging. Permitted values are {li lo ma mn xp ex ss fe ba qc qe wq re tr fs fc mc md mr ms mu}

21.68 actionattributes Option

The `actionattributes` option (available for `imap`, `pop`, and `messagetrace`) specifies the action attributes enabled in Message Store transaction logging. Permitted values are {ts pi si us mi ii tr if fi sz no tg ac ct ma om ut ua bs dq du ev mm mc sq sn un ur uo at cs cn cc}

21.69 enableuserlist Option Under imap

The `enableuserlist` IMAP option enables `imsconnutil` connected user listing for IMAP service.

21.70 forcetelemetry Option Under imap

Setting the `forcetelemetry` IMAP option to 1 forces telemetry for all users. Warning: this generates a lot of data and should not be used on a production system.

See also the [logcommands](#) IMAP option.

21.71 idletimeout Option Under imap

The `idletimeout` IMAP option specifies a maximum idle time, in minutes, for IMAP connections.

21.72 logprotocolerrors Option Under imap

If the `logprotocolerrors` IMAP option is greater than zero, protocol errors are logged as debug messages for IMAP.

21.73 logauthsessionid Option

Set the `logauthsessionid` IMAP option to include a numeric session id in square brackets at the end of protocol authentication responses. This can be used to correlate authentication errors in the log with authentication errors sent over the network.

21.74 maxprotocolerrors Option Under imap

The `maxprotocolerrors` IMAP option specifies the maximum number of protocol errors allowed before the IMAP connection is forcibly closed.

21.75 maxsessions Option Under imap

The `maxsessions` IMAP option specifies the maximum number of sessions per IMAP server process.

21.76 maxthreads Option Under imap

The `maxthreads` IMAP option specifies the maximum number of threads per IMAP server process.

21.77 numprocesses Option Under imap

the `numprocesses` IMAP option specifies the number of IMAP server processes. Note that the Watcher must be enabled for stop-msg to correctly shut down all processes if this is set to a value larger than one.

21.78 plaintextmncipher Option Under imap

If the `imap.plaintextmncipher` option 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.

21.79 sslcachesize Option Under imap

The `sslcachesize` IMAP option specifies the number of SSL sessions to be cached by the IMAP server.

21.80 sslnicknames Option Under imap

The `sslnicknames` IMAP option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for IMAP to offer clients if SSL/TLS enabled. Overrides for IMAP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

21.81 sslport Option Under imap

The `sslport` IMAP option specifies the port number for the IMAP over SSL service. The default is 993. Note that to enable the IMAP+SSL service, the `enablesslport` and `sslusessl` IMAP options must be set.

When using the MMP (IMAP Proxy), see its `sslbacksideport` option to tell the IMAP Proxy to attempt to connect with SSL to the IMAP `sslport`.

21.82 sslusessl Option Under imap

If a server certificate is installed and the `sslusessl` IMAP option is not set to 0, then STARTTLS is enabled on the IMAP server (listening at its regular [port](#)).

As regards listening at a separate `sslport`, note that for the 7.0.5 release, the `sslusessl` option must be *explicitly* set to 1 (even though the default was 1) as well as setting [enablenesslport](#) to enable SSL connections on a separate `sslport`. For the 8.0 release, it is no longer necessary to explicitly set this option in order to enable SSL connections on a separate port.

21.83 logunauthsession Option Under imap

The `logunauthsession` IMAP option enables log messages from unauthenticated client IMAP sessions. Prior to turning this on, consider verifying that your logging filesystem can handle the amount of I/O possible from unauthenticated clients connecting frequently.

21.84 IMAP password expiration alert options

A few `pwexpirealert` options under `imap` control the sending of IMAP ALERT notifications to IMAP users to warn that the user's password will expire.

21.84.1 firstwarn Option

Setting the `firstwarn` IMAP password expiration alert option, `imap.pwexpirealert.firstwarn`, causes sending an IMAP ALERT to notify a user that their password will expire soon. The value specifies the number of seconds of remaining password validity before a warning is sent. For example, specify 259200 ($3*24*60*60$) to begin sending warnings 3 days before expiration.

21.84.2 viametermaid Option

By default the IMAP server limits password expirations to once per day on a per-process basis. Set the `viametermaid` IMAP password expiration alert option, `imap.pwexpirealert.viametermaid`, to use [MeterMaid](#) to get per-metermaid instance limits instead. If this is set, then it is also necessary to set at least the MeterMaid [secret](#) option (`metermaid.config.secret` in legacy configuration, or in Unified Configuration [metermaid.secret](#) or alternatively `mta.metermaid_secret`), as well as other relevant [MeterMaid client](#) settings.

21.84.3 metermaidtable Option

The `metermaidtable` IMAP password expiration alert option, `imap.pwexpirealert.metermaidtable`, specifies the name of the [MeterMaid](#) table to use for password expiration alerts in IMAP.



Chapter 22 POP options

22.1 enable Option Under pop	22-1
22.2 port Option Under pop	22-1
22.3 poplogmbxstat Option	22-2
22.4 popstatuskludge Option	22-2
22.5 lockmailbox Option	22-2
22.6 emulategpopper Option	22-2
22.7 allowanonymouslogin Option Under pop	22-2
22.8 banner Option Under pop	22-2
22.9 domainallowed Option Under pop	22-2
22.10 domainnotallowed Option Under pop	22-2
22.11 enablenesslport Option Under pop	22-2
22.12 forcetelemetry Option Under pop	22-3
22.13 idletimeout Option Under pop	22-3
22.14 logprotocolerrors Option Under pop	22-3
22.15 maxprotocolerrors Option Under pop	22-3
22.16 maxsessions Option Under pop	22-3
22.17 maxthreads Option Under pop	22-3
22.18 numprocesses Option Under pop	22-3
22.19 plaintextmncipher Option Under pop	22-3
22.20 sslcachesize Option Under pop	22-3
22.21 sslnicknames Option Under pop	22-4
22.22 sslport Option Under pop	22-4
22.23 sslusessl Option Under pop	22-4
22.24 logunauthsession Option Under pop	22-4

The POP server has a number of options.

See also the following options which are described rather generically under [Base options](#), but which may also be set specifically under `pop` (as for instance if one wishes to have POP use a different value than the general base value): the various `bg*` options, and `defaultdomain`.

Note that `msprobe` can probe for whether the POP server is running; see `msprobe's` `msprobe.probe:pop` options.

22.1 enable Option Under pop

The `enable` POP option, (`pop.enable` in Unified Configuration, or `service.pop.enable` in legacy configuration), enables the POP service on `start-msg` startup. Note: POP over SSL service is enabled/disabled separately using `pop.enablenesslport` in Unified Configuration, or `service.pop.enablenesslport` in legacy configuration.

This option defaults to 0 if not set, but initial configuration may enable the option as appropriate.

22.2 port Option Under pop

The `port` POP option specifies the POP server port number.

22.3 poplogmbxstat Option

The `poplogmbxstat` POP option, if set to 1 (the default being 0), causes the POP log to show mailbox statistics on login and logout.

22.4 popstatuskludge Option

RESTRICTED: The `popstatuskludge` POP option, if set to 1 (the default of this restricted option being 0), enables the POP server to generate a message Status: header line on the fly indicating messages as unread or read based upon saving the highest message read by the client.

22.5 lockmailbox Option

When set to 1 (on), the `lockmailbox` POP option limits the number of POP sessions allowed to access a mailbox at a time to one. When set to 0 (off), POP users can access mailboxes in multiple sessions concurrently.

22.6 emulateqpopper Option

Hack.

22.7 allowanonymouslogin Option Under pop

The `allowanonymouslogin` POP option sets whether or not anonymous login is allowed by POP.

22.8 banner Option Under pop

The `banner` POP option specifies the POP protocol welcome banner. One line string, with virtual parameters: %h=hostname, %p=protocol(ESMTP,POP or IMAP), %P=Product Name, %v and %V=Version (short or long).

22.9 domainallowed Option Under pop

The `domainallowed` POP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed POP access.

22.10 domainnotallowed Option Under pop

The `domainnotallowed` POP option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed POP access.

22.11 enablesslport Option Under pop

The `enablesslport` POP option sets whether or not POP over SSL service is started; if enabled, this service uses the port set in the `sslport` POP option. For the 7.0.5 release, the

`sslusessl` option must also be explicitly set to enable the separate SSL port. For the 8.0 release, setting this option enables the separate SSL port and it is no longer necessary to explicitly set the `sslusessl` option.

22.12 forcetelemetry Option Under pop

Setting the `forcetelemetry` POP option to 1 forces telemetry for all users. Warning: this generates a lot of data and should not be used on a production system.

22.13 idletimeout Option Under pop

The `idletimeout` POP option specifies a maximum idle time, in minutes, for POP connections.

22.14 logprotocolerrors Option Under pop

If the `logprotocolerrors` POP option is greater than zero, protocol errors are logged as debug messages for POP.

22.15 maxprotocolerrors Option Under pop

The `maxprotocolerrors` POP option specifies the maximum number of protocol errors allowed before the POP connection is forcibly closed.

22.16 maxsessions Option Under pop

The `maxsessions` POP option specifies the maximum number of sessions per server process.

22.17 maxthreads Option Under pop

The `maxthreads` POP option specifies the maximum number of threads per POP server process.

22.18 numprocesses Option Under pop

The `numprocesses` POP option specifies the number of POP server processes. Note that the Watcher must be enabled for `stop-msg` to correctly shut down all processes if this is set to a value larger than one.

22.19 plaintextmincipher Option Under pop

If the `pop.plaintextmincipher` option 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.

22.20 sslcachesize Option Under pop

The `sslcachesize` POP option specifies the number of SSL sessions to be cached by the POP server.

22.21 sslnicknames Option Under pop

The `sslnicknames` POP option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for POP to offer clients if SSL/TLS enabled. Overrides for POP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

22.22 sslport Option Under pop

The `sslport` POP option specifies the port number for the POP over SSL port. The default is 995. Note that to enable the POP+SSL service, the `enablesslport` and `sslusessl` POP options must be set.

When using the MMP (POP Proxy), see its `sslbacksideport` option to tell the POP Proxy to attempt to connect with SSL to the POP `sslport`.

22.23 sslusessl Option Under pop

If a server certificate is installed and the `sslusessl` POP option is not set to 0, then STARTTLS is enabled on the POP server (listening at its regular `port`).

As regards listening at a separate `sslport`, note that for the 7.0.5 release, the `sslusessl` option must be *explicitly* set to 1 (even though the default was 1) as well as setting `enablesslport` to enable SSL connections on a separate `sslport`. For the 8.0 release, it is no longer necessary to explicitly set this option in order to enable SSL connections on a separate port.

22.24 logunauthsession Option Under pop

The `logunauthsession` POP option enables log messages from unauthenticated client POP sessions. Prior to turning this on, consider verifying that your logging filesystem can handle the amount of I/O possible from unauthenticated clients connecting frequently.

Chapter 23 Message Trace options

23.1 activate Option	23-1
23.2 actions Option	23-1
23.3 actionattributes Option	23-1
23.4 loglevel Option Under messagetrace	23-1

Message Trace can be enabled by setting the `messagetrace.activate` option to `yes`. The `actions` and `actionattributes` options may be used to further control the detail included in message tracing. The only other Message Trace options are `logfile options` set as `messagetrace.logfile.*`.

23.1 activate Option

The `activate` Message Trace option, `messagetrace.activate` (or in legacy configuration the `configutil` parameter `local.msgtrace.active`), enables message tracing. Permitted values are `yes`, `msgtrace`, and `transactlog`.

23.2 actions Option

The `actions` option (available for `imap`, `pop`, and `messagetrace`) specifies the actions enabled in Message Store transaction logging. Permitted values are `{li lo ma mn xp ex ss fe ba qc qe wq re tr fs fc mc md mr ms mu}`

23.3 actionattributes Option

The `actionattributes` option (available for `imap`, `pop`, and `messagetrace`) specifies the action attributes enabled in Message Store transaction logging. Permitted values are `{ts pi si us mi ii tr if fi sz no tg ac ct ma om ut ua bs dq du ev mm mc sq sn un ur uo at cs cn cc}`

23.4 loglevel Option Under messagetrace

Message Trace data is inherently information level. So setting the `messagetrace.loglevel` option to a higher value than `information` will suppress the recording of Message Trace data.



Chapter 24 notifytarget options

24.1 enable Option Under notifytarget	24-2
24.2 notifytype Option	24-2
24.3 enseventkey Option Under notifytarget	24-2
24.4 enshost Option Under notifytarget	24-2
24.5 ensport Option Under notifytarget	24-2
24.6 jmghost Option Under notifytarget	24-2
24.7 jmgsport Option Under notifytarget	24-2
24.8 jmgspwd Option Under notifytarget	24-3
24.9 jmgsopic Option Under notifytarget	24-3
24.10 jmgsuser Option Under notifytarget	24-3
24.11 jmgsqueue Option	24-3
24.12 maxbodysize Option Under notifytarget	24-3
24.13 maxheadersize Option Under notifytarget	24-3
24.14 msgflags Option Under notifytarget	24-3
24.15 destinationtype Option	24-4
24.16 ldapdestination Option	24-4
24.17 persistent Option	24-4
24.18 priority Option	24-4
24.19 ttl Option	24-4
24.20 deletemsg Option Under notifytarget	24-4
24.21 loguser Option Under notifytarget	24-4
24.22 newmsg Option Under notifytarget	24-4
24.23 overquota Option Under notifytarget	24-5
24.24 underquota Option Under notifytarget	24-5
24.25 setacl Option	24-5
24.26 noninbox Option Under notifytarget	24-5
24.27 msgtypes Option	24-5
24.28 purgmsg Option Under notifytarget	24-5
24.29 readmsg Option Under notifytarget	24-5
24.30 updatemsg Option Under notifytarget	24-5
24.31 expungemsg Option	24-6
24.32 annotatemsg Option	24-6
24.33 changeflag Option	24-6
24.34 copymsg Option	24-6

Named notifytarget groups are used to control the operation of sending notifications to an [ENS server](#) or JMQ broker. Such notifytarget groups replace the legacy configuration `local.store.notifyplugin.*.* configutil` options.

For instance, to configure notifications to an ENS server (whose basic options are shown in the example below), one makes a named notifytarget group, in this example `ens1`, and sets any necessary options below that:

```
msconfig> show ens.*
role.ens.enable = 1
msconfig> show -default base.listenaddr
base.listenaddr: <no-default>
msconfig> show -default ens.port
ens.port: 7997
```

```
msconfig> set notifytarget:ens1.enable 1
msconfig# set notifytarget:ens1.enshost 127.0.0.1
msconfig# set notifytarget:ens1.ensport 7997
```

24.1 enable Option Under notifytarget

The `enable` option under `notifytarget`, `notifytarget:target-name.enable`, specifies whether to generate notifications for this destination. This defaults to 1. This can also be used to disable the default destination for a JMQ plugin, so notifications are only sent to destinations specified in the `ldapdestination` attribute in users' directory entries.

24.2 notifytype Option

The `notifytype` option under `notifytarget` specifies the type of this notification target. Presently the value can be either `ens` (the default) or `jmj`. Note that support for `jmj` is deprecated and will be removed in a future release. For XML mode this setting is used in each `notifytarget` instead of having one `"local.store.notifyplugin"` string that lists all the plugins.

24.3 enseventkey Option Under notifytarget

The `enseventkey` option under `notifytarget`, `notifytarget:named-target.enseventkey`, specifies the event key to use for ENS notifications. The default is `"enp://127.0.0.1/store/%M"` where `%M` is replaced at runtime with the mailbox name for mailbox-related events. The value of the `enseventkey` option must not contain any question mark `"?"` characters.

24.4 enshost Option Under notifytarget

The `enshost` option under `notifytarget`, `notifytarget:target-name.enshost`, specifies the IP address or hostname of the ENS server. If unset, defaults to the value of the `listenaddr` base option (`service.listenaddr` in legacy configuration) or the loopback address.

24.5 ensport Option Under notifytarget

The `ensport` option under `notifytarget`, `notifytarget:target-name.ensport`, specifies the TCP port number of the ENS server. This will generally correspond to the setting of the `ens.port` option (`local.ens.port` in legacy configuration).

24.6 jmjhost Option Under notifytarget

The `jmjhost` option under `notifytarget`, `notifytarget:target-name.jmjhost`, specifies the hostname of the JMQ broker.

24.7 jmjport Option Under notifytarget

The `jmjport` option under `notifytarget`, `notifytarget:target-name.jmjport`, specifies the port number of the JMQ broker. The default is 7676.

24.8 jmqpwd Option Under notifytarget

The `jmqpwd` option under `notifytarget`, `notifytarget:target-name.jmqpwd`, specifies the Glassfish MQ (formerly called Java MQ or JMQ) user password that is used to authenticate to the Glassfish MQ broker. The default value was removed in the 8.0 release. For security reasons, this should not be set to an easy-to-guess value such as `guest`.

24.9 jmqtopic Option Under notifytarget

The `jmqtopic` option under `notifytarget`, `notifytarget:target-name.jmqtopic`, specifies the name of the topic or queue to which JMQ will publish events. The default is `JES-MS`. Note that if `notifytarget:target-name.jmqqueue` is set, it will override the `jmqtopic` value.

Note that the corresponding `target-name` setting of `notifytarget:target-name.destinationtype` is what controls whether in fact a topic *vs.* queue is used.

24.10 jmquser Option Under notifytarget

The `jmquser` option under `notifytarget`, `notifytarget:target-name.jmquser`, specifies the JMQ username. The default is `"guest"`.

24.11 jmqqueue Option

The `jmqqueue` option under `notifytarget`, `notifytarget:target-name.jmqqueue`, specifies the name of the topic or queue to which JMQ will publish events; (if set, this overrides `jmqtopic`, or in legacy configuration `local.store.notifyplugin.*.jmqtopic`).

Note that the corresponding `target-name` setting of `notifytarget:target-name.destinationtype` is what controls whether in fact a topic *vs.* queue is used.

24.12 maxbodysize Option Under notifytarget

The `maxbodysize` option under `notifytarget`, `notifytarget:target-name.maxbodysize`, specifies the maximum size (in bytes) of the body that will be transmitted with the notification.

24.13 maxheadersize Option Under notifytarget

The `maxheadersize` option under `notifytarget`, `notifytarget:target-name.maxheadersize`, specifies the maximum size (in bytes) of the header that will be transmitted with the notification.

24.14 msgflags Option Under notifytarget

The `msgflags` option under `notifytarget`, `notifytarget:named-target.msgflags`, enables the `msgflag` notification mechanism.

24.15 destinationtype Option

The `destinationtype` option under `notifytarget`, `notifytarget:target-name.destinationtype`, specifies the JMQ destination type, `queue` or `topic` (the default).

24.16 ldapdestination Option

The `ldapdestination` option under `notifytarget`, `notifytarget:target-name.ldapdestination`, specifies what LDAP attribute is to be used to look up a JMQ notification destination. If this is not specified, or the user lacks this attribute, or the value of this attribute is zero length, the library id is used as the destination. Although any attribute can be used, the `mailEventNotificationDestination` attribute has been defined for this purpose.

24.17 persistent Option

The `persistent` option under `notifytarget`, `notifytarget:target-name.persistent`, specifies whether persistent JMQ messages are to be used. The default of 0 means that non-persistent JMQ messages are used.

24.18 priority Option

The `priority` option under `notifytarget`, `notifytarget:target-name.priority`, specifies the priority to be used for JMQ notification messages.

24.19 ttl Option

The `ttl` option under `notifytarget`, `notifytarget:target-name.ttl`, specifies the time-to-live (in milliseconds) for JMQ messages. 0, the default, means no timeout.

24.20 deletemsg Option Under notifytarget

The `deletemsg` option under `notifytarget`, `notifytarget:target-name.deletemsg`, specifies whether `DeleteMsg` events will generate a notification.

24.21 loguser Option Under notifytarget

The `loguser` option under `notifytarget`, `notifytarget:target-name.loguser`, specifies whether `LogUser` events will generate a notification. For JMQ notifications (`notifytarget:target-name.notifytype=jmq`), the default is 0. For ENS notifications (`notifytarget:target-name.notifytype=ens`), the default is 1 for the `ms-internal` target, but 0 for other `target-name` targets.

24.22 newmsg Option Under notifytarget

The `newmsg` option under `notifytarget`, `notifytarget:target-name.newmsg`, specifies whether NewMsg events will generate a notification.

24.23 overquota Option Under notifytarget

The `overquota` option under `notifytarget`, `notifytarget:target-name.overquota`, specifies whether OverQuota events will generate a notification.

24.24 underquota Option Under notifytarget

The `underquota` option under `notifytarget`, `notifytarget:target-name.underquota`, specifies whether UnderQuota events will generate a notification.

24.25 setacl Option

The `setacl` option under `notifytarget`, `notifytarget:target-name.setacl`, specifies whether SetAcl events will generate a notification.

24.26 noninbox Option Under notifytarget

The `noninbox` option under `notifytarget`, `notifytarget:target-name.noninbox`, determines whether all folders generate notifications or if only the INBOX generates notifications: 0: only INBOX, 1: all folders. As of 7 Update 4, this defaults to 1 for ENS and 0 for JMQ. As of Messaging Server 7.0.5, this defaults to 1 for both ENS and JMQ.

24.27 msgtypes Option

For JMQ notifications (`notifytarget:target-name.notifytype=jmq`), the `msgtypes` option under `notifytarget`, `notifytarget:target-name.msgtypes`, determines whether to include message type counts in [PurgeMsg](#), [DeleteMsg](#), [CopyMsg](#), [NewMsg](#), and [UpdateMsg](#) messages.

24.28 purgemsg Option Under notifytarget

The `purgemsg` option under `notifytarget`, `notifytarget:target-name.purgemsg`, specifies whether PurgeMsg events will generate a notification.

24.29 readmsg Option Under notifytarget

The `readmsg` option under `notifytarget`, `notifytarget:target-name.readmsg`, specifies whether ReadMsg events will generate a notification.

24.30 updatemsg Option Under notifytarget

The `updatemsg` option under `notifytarget`, `notifytarget:target-name.updatemsg`, specifies whether UpdateMsg events will generate a notification.

24.31 expungemsg Option

The `expungemsg` option under `notifytarget`, `notifytarget:target-name.expungemsg`, specifies whether `ExpungeMsg` events will generate a notification.

24.32 annotatemsg Option

The `annotatemsg` option under `notifytarget`, `notifytarget:target-name.annotatemsg`, specifies whether `AnnotateMsg` events will generate a notification.

24.33 changeflag Option

The `changeflag` option under `notifytarget`, `notifytarget:target-name.changeflag`, specifies whether `ChangeFlag` events will generate a notification. For `JMQ` notifications (`notifytarget:target-name.notifytype=jmq`), the default is 0. For `ENS` notifications (`notifytarget:target-name.notifytype=ens`), the default is 1.

24.34 copymsg Option

The `copymsg` option under `notifytarget`, `notifytarget:target-name.copymsg`, specifies whether `IMAP COPY` operations will generate `Copy` events or `UpdateMsg` events.

Chapter 25 Indexer options

25.1 <code>enable</code> Option Under indexer	25-1
25.2 <code>port</code> Option Under indexer	25-1
25.3 <code>server_host</code> Option Under indexer	25-2
25.4 <code>timeout</code> Option Under indexer	25-2
25.5 <code>connecttimeout</code> Option Under indexer	25-2
25.6 <code>bodytextonly</code> Option	25-2
25.7 <code>substring_search</code> Option	25-2
25.8 <code>suffix_search</code> Option	25-2
25.9 <code>prefix_search</code> Option	25-3

Enabling the Messaging Server's ISS client (via `imap.indexer.enable=1` and a valid setting for the `imap.indexer.server_host` option) causes certain IMAP search and sort commands to be sent to ISS (the Indexing and Search Service). See the `imap.indexer.enable` description for more details.

There are several Indexer options further affecting Messaging Server's consultations of ISS. Also see the `indexeradmins` Message Store option, `store.indexeradmins`.

25.1 `enable` Option Under indexer

The `enable` Indexer option, if set to 1, causes sending some IMAP search and sort commands to ISS, the Indexing and Search Service. The `server_host` Indexer option must also be set for this to function correctly. The ISS processes search operations based on words rather than substrings so results may not be IMAP standards compliant and may differ from a search performed by the IMAP server.

If the ESEARCH RETURN (ALL) result option is present in the search command, then ISS is used. For all other ESEARCH extension features, the ISS is not used. The ISS is not used if KEYWORD, HEADER, OLDER, YOUNGER, MODSEQ, ANNOTATION, or RECENT appear. At least one of the following search terms: SUBJECT, FROM, TO, CC, BCC, TEXT or BODY must be present for the ISS to be used. If an error occurs from the ISS, then the search may fall back to processing by the IMAP server. See the ISS documentation for details on what searches it supports as it may not support all combinations of AND, OR and NOT operators that IMAP supports. The specific rules of what is sent to the ISS and what is processed locally on the IMAP server may change in future releases.

The `bodytextonly` option modifies the above rules so that TEXT or BODY must be present or the search will not be sent to the ISS.

As of Messaging Server 7.0.5.30.0, the ISS is used to process IMAP ESEARCH commands with RETURN (ALL) result option. For all other ESEARCH extension features, the ISS is not used.

Also see the `debugkeys` option's search key to enable IMAP search and sort command related debug.

25.2 `port` Option Under indexer

The `port` Indexer option specifies the TCP port on which ISS listens for incoming TCP connections, *i.e.*, the TCP port to which Messaging Server should connect to communicate with ISS.

If the `indexer.sslusessl` option (`service.imap.indexer.sslusessl` option in legacy configuration) is set, the IMAP server uses SSL to authenticate to ISS on this port.

25.3 server_host Option Under indexer

The `server_host` Indexer option specifies the fully qualified host name or IP address where the Indexing and Search Server runs. This `indexer.server_host` option must be set, to tell Messaging Server where ISS is running, if the IMAP server has been told via `imap.indexer.enable=1` to consult ISS.

25.4 timeout Option Under indexer

The `timeout` Indexer option specifies the timeout in seconds for read and write operations between the IMAP server and ISS.

25.5 connecttimeout Option Under indexer

The `connecttimeout` `indexer` option, (`indexer.connecttimeout` in Unified Configuration, or `service.imap.indexer.connectwait` in legacy configuration), specifies how long in seconds the IMAP server should wait for a connection to be established to the Indexing and Search Service (ISS). Attempting to set this option to a value greater than 30 will result in a value of 10 being used.

25.6 bodytextonly Option

If the `bodytextonly` Indexer option is set to 1, then send IMAP search queries to ISS, the Indexing and Search Server, if the query contains BODY or TEXT terms and the query does not contain any search terms that ISS does not support.

25.7 substring_search Option

The `substring_search` Indexer option specifies which, if any, search parameters sent to ISS, the Indexing and Search Server, should be preceded and followed by a "*", so that a search for `subject ear` would match "searches" as well as "ear". The search parameter will not be preceded and followed by the '*' character, if it contains a space character or a double quote character or an opening parentheses in the beginning or a closing parenthesis at the end. The value given should be one or more of `subject text body from to cc bcc` separated by one or more spaces. Use with caution as this imposes a significant load on ISS.

25.8 suffix_search Option

The `suffix_search` Indexer option specifies which, if any, search parameters sent to ISS, the Indexing and Search Server, should be followed by a "*", so that a search for `subject search` would match "searches" as well as "search". The search parameter will not be followed by the '*' character, if it contains a space character or a double quote character or an opening parentheses in the beginning or a closing parenthesis at the end. The value given should be one or more of `subject text body from to cc bcc` separated by one or more spaces.

(Note that the meanings of `suffix_search` and `prefix_search` could be considered reversed; read carefully before setting.)

25.9 prefix_search Option

The `prefix_search` Indexer option specifies which, if any, search parameters sent to ISS, the Indexing and Search Server, should be preceded by a "*", so that a search for `subject mine` would match "determine" as well as "mine". The search parameter will not be preceded by the '*' character, if it contains a space character or a double quote character or a opening parentheses in the beginning or a closing parenthesis at the end. The value given should be one or more of `subject text body from to cc bcc` separated by one or more spaces. Use with caution as this imposes a significant load on ISS.

(Note that the meanings of `suffix_search` and `prefix_search` could be considered reversed; read carefully before setting.)



Part IV Proxies and the MMP

Proxies and the MMP, for accessing the Message Store (and also an SMTP Proxy for use by local clients) have a number of configuration options.



Chapter 26 Proxy options

26.1 httpadminpass Option	26-1
26.2 imapadmin Option	26-1
26.3 imapadminpass Option	26-1
26.4 imapport Option	26-1
26.5 storehostlist Option	26-2

Various options may be set under a named proxy group to control aspects of proxy connection authentication and port and hosts. Note that since such options are set under a named proxy group, where the group name is a host name, such options are set using syntax such as:

```
msconfig> set proxy:host\domain.com.proxy-option-name option-value
```

(In particular, note that when a group name has embedded period characters, as is routine for hostnames, each such embedded period in the name requires backslash quoting on the command line.)

See also the [base options](#) with names beginning proxy* as they set defaults if the proxy options are not set.

26.1 httpadminpass Option

DEPRECATED: See [proxyadminpass](#).

The httpadminpass Proxy option specifies the store admin password for a specific host if different from local.service.http.proxy.adminpass. Not configured by default.

26.2 imapadmin Option

The imapadmin proxy option specifies the store admin login name for a specific host if different from [proxyadmin](#) base option (local.service.proxy.admin in legacy configuration). Not configured by default.

26.3 imapadminpass Option

The imapadminpass proxy option specifies a store admin password for a specific host if different from [proxyadminpass](#) (local.service.proxy.adminpass in legacy configuration). Not configured by default.

26.4 imapport Option

The imapport proxy option (legacy configuration local.service.proxy.imapport.hostname) specifies the IMAP port number used when connecting to the backend mail store hostname specified by the Unified Configuration group name for the proxy group. Thus a setting of imapport could appear along the lines of:

```
msconfig> set proxy:host\.domain\.com.imapport 143
```

where each period character in the backend host name `host.domain.com` must be quoted with a backslash character in the `msconfig` command. If `imapport` is set, then it overrides for connections to that backend mail store the `base.proxyimapport` option's value (default 143).

26.5 storehostlist Option

The `storehostlist` Proxy option specifies a list of server hostnames that have access to the same message store data. The first hostname in the list should be the default master for the message store while the other hosts in the list should be configured to act as failover hosts for the master. When a `mailHost` attribute is found in LDAP, it is first resolved via this configuration. This allows MMP, `mshttpd`, `imapd` and MTA servers to try connecting to the mail store via an alternative host. Note that support for this by the ENS client is presently missing (but not needed for IMAP IDLE to work correctly). The hostnames used in this option must match the hostnames in the `mailHost` attributes exactly (although ASCII case variations are permitted).

Chapter 27 MMP and IMAP Proxy and POP Proxy and SUBMIT Proxy and vdomain options

27.1 enable Option Under the MMP	27-5
27.2 authcachettl Option	27-5
27.2.1 Use with base	27-5
27.3 authenticationldapattributes Option	27-6
27.4 authenticationserver Option	27-6
27.5 authservice Option	27-6
27.6 authservicettl Option	27-6
27.7 backsideport Option	27-6
27.8 banner Option Under the MMP	27-6
27.9 banner Option Under the IMAP proxy	27-7
27.10 banner Option Under the POP proxy	27-7
27.11 banner Option Under the submit proxy	27-7
27.12 bethegroup Option	27-7
27.13 betheuser Option	27-7
27.14 bgmax Option	27-7
27.15 bgpenalty Option	27-8
27.16 bgmaxbadness Option	27-8
27.17 bgdecay Option	27-8
27.18 bglinear Option	27-8
27.19 bgexcluded Option	27-8
27.20 binddn Option	27-8
27.21 bindpass Option	27-8
27.22 canonicalvirtualdomainindelim Option	27-9
27.23 capability Option	27-9
27.24 certmapdn Option	27-9
27.25 certmapfile Option	27-9
27.26 clientlookup Option	27-9
27.27 connecttimeout Option Under the MMP	27-10
27.28 connecttimeout Option Under the IMAP proxy	27-10
27.29 connecttimeout Option Under the POP proxy	27-10
27.30 connlimits Option	27-10
27.31 connrejectthreshold Option	27-11
27.32 crams Option	27-11
27.33 debugkeys Option	27-11
27.33.1 Use with base	27-12
27.34 defaultdomain Option	27-12
27.34.1 Use with base	27-13
27.35 domainallowed Option	27-13
27.35.1 Use with http	27-13
27.35.2 Use with imap	27-13
27.35.3 Use with pop	27-13
27.35.4 Use with ens	27-13
27.35.5 Use with eval_ldapd	27-13
27.36 domainnotallowed Option	27-13
27.36.1 Use with eval_ldapd	27-13

27.36.2 Use with http	27-14
27.36.3 Use with imap	27-14
27.36.4 Use with pop	27-14
27.36.5 Use with ens	27-14
27.37 domainsearchformat Option	27-14
27.38 ehlokeywords Option	27-14
27.39 failovertimeout Option	27-15
27.40 hosteddomains Option	27-15
27.41 ipv6in Option	27-15
27.42 ipv6out Option	27-15
27.43 langlist Option	27-16
27.44 ldapcachesize Option	27-16
27.45 ldapcachettl Option	27-16
27.46 ldappendingoplmit Option	27-16
27.47 ldaprefreshinterval Option	27-16
27.48 ldaptimeout Option	27-17
27.49 ldapurl Option	27-17
27.50 logdir Option	27-17
27.51 loglevel Option Under the MMP	27-17
27.52 loglevel Option Under the IMAP proxy	27-18
27.53 loglevel Option Under the POP proxy	27-18
27.54 loglevel Option Under the submit proxy	27-18
27.55 mailhostattrs Option	27-18
27.56 maxconcurrentconnectionattempts Option	27-18
27.57 maxthreads Option Under the MMP	27-18
27.58 numprocesses Option Under the MMP	27-18
27.59 numthreads Option	27-19
27.60 plaintextmncipher Option	27-19
27.60.1 Use with http	27-19
27.60.2 Use with imap	27-19
27.60.3 Use with pop	27-19
27.60.4 Use with mta	27-19
27.61 polldelay Option	27-19
27.62 popbeforesmtpkludgchannel Option	27-20
27.63 preauth Option	27-20
27.64 preauthtimeout Option	27-20
27.65 preferpoll Option	27-20
27.66 replayformat Option	27-20
27.66.1 Use with http	27-21
27.67 replaypass Option	27-21
27.68 requireauthenticationserver Option	27-21
27.69 restrictplainpasswords Option	27-21
27.70 searchformat Option	27-21
27.71 serverdownalert Option	27-22
27.72 servicelist Option	27-22
27.73 smtpproxypassword Option	27-22
27.74 smtprelays Option	27-23
27.75 spoofemptymailbox Option	27-23
27.76 spooftempfail Option	27-23
27.77 spoofmessagefile Option	27-23
27.78 ssladjustciphersuites Option	27-23
27.79 sslbacksideport Option	27-25
27.80 sslcachedir Option	27-25

27.81	sslcertprefix Option	27-25
27.82	sslkeypasswdfile Option	27-26
27.83	sslkeyprefix Option	27-26
27.84	sslnicknames Option	27-26
27.84.1	Use with base	27-26
27.84.2	Use with http	27-26
27.84.3	Use with imap	27-26
27.84.4	Use with mta	27-27
27.84.5	Use with pop	27-27
27.84.6	Use with ens	27-27
27.85	sslsecmodfile Option	27-27
27.86	storeadmin Option	27-27
27.87	storeadminpass Option	27-27
27.88	syncldap Option	27-27
27.89	tcpaccess Option	27-28
27.90	tcpaccessattr Option	27-28
27.91	timeout Option Under the MMP	27-28
27.92	timeout Option Under the IMAP proxy	27-28
27.93	timeout Option Under the POP proxy	27-28
27.94	timeout Option Under the submit proxy	27-28
27.95	use_nslog Option	27-28
27.95.1	Use with dispatcher	27-29
27.95.2	Use with job controller	27-29
27.96	usenslog Option	27-29
27.97	usergroupdn Option	27-29
27.98	virtualdomaindelim Option	27-29
27.99	virtualdomainfile Option	27-29

There are many options affecting MMP operation, or operation of its subcomponents the IMAP Proxy, the POP Proxy, or the SUBMIT Proxy, plus additional options modifying the support for "virtual domains". These options are often available for setting at more than one scope; they are listed, with their defaults, in [MMP and its subcomponents, available options and their defaults](#).

Table 27.1 MMP and its subcomponents, available options and their defaults

Option	MMP	IMAP Proxy	POP Proxy	SUBMIT Proxy	vdomain
authcachettl	✓ 900	✓ 900	✓ 900		✓ 900
authenticationldapattributes		✓	✓		✓
authenticationserver		✓	✓		
authservice			✓ 0		✓ 0
authservicettl			✓ -1		✓ -1
backsideport		✓ 143	✓ 110		
banner	✓ <i>complex</i>	✓ <i>complex</i>	✓ <i>complex</i>		
bethesgroup: deleted; see user in restricted.cnf instead					
betheuser: deleted; see user in restricted.cnf instead					
bgdecay	✓ 900	✓ 900	✓ 900		
bgexcluded	✓	✓	✓		
bglinear	✓ 0	✓ 0	✓ 0		
bgmax	✓ 10000	✓ 10000	✓ 10000		
bgmaxbadness	✓ 60	✓ 60	✓ 60		
bgpenalty	✓ 2	✓ 2	✓ 2		
binddn : deprecated; see base.ugldapbinddn instead		✓	✓	✓	✓

bindpass : deprecated; see base.ugldapbindcred instead		✓	✓	✓	✓
canonicalvirtualdomaindelim	✓ @	✓ @	✓ @		
capability		✓ <i>very long list---see text</i>			
certmapdn : alias for usergroupdn					
certmapfile : deleted					
clientlookup				✓ 0	✓ 0
connecttimeout	✓ 30	✓ 30	✓ 30		
connlimits	✓ :20	✓	✓	✓	
connrejectthreshold	✓ <i>complex---see text</i>				
crams	✓ 0	✓ 0	✓ 0		✓ 0
debugkeys	✓	✓	✓	✓	✓
defaultdomain	✓	✓	✓	✓	✓
domainallowed		✓	✓	✓	
domainnotallowed		✓	✓	✓	
domainsearchformat	✓ (uid=%U)	✓ (uid=%U)	✓ (uid=%U)		✓ (uid=%U)
ehlokeywords				✓	✓
enable	✓ 0				
failovertimeout				✓ 10	✓ 10
hosteddomains	✓ 1	✓ 1	✓ 1		✓ 1
ipv6in	✓ 0				
ipv6out	✓ 0				
ipv6sortorder	✓ default				
langlist	✓ i-default EN	✓ i-default EN			
ldapcachesize	✓ <i>complex</i>	✓ <i>complex</i>	✓ <i>complex</i>	✓ <i>complex</i>	✓ <i>complex</i>
ldapcachettl	✓ 900		✓ 900	✓ 900	✓ 900
ldappendingoplimit	✓ 128	✓ 128	✓ 128		
ldaprefreshinterval	✓ 2100	✓ 2100	✓ 2100		
ldaptimeout : deprecated	✓ 60	✓ 60	✓ 60		
ldapurl : deprecated	✓ ldap://localhost/o=internet	✓ ldap://localhost/o=internet	✓ ldap://localhost/o=internet	✓ ldap://localhost/o=internet	
logdir	✓	✓	✓	✓	
loglevel	✓ notice	✓ notice	✓ notice	✓ notice	
mailhostattns	✓ mailHost	✓ mailHost	✓ mailHost		✓ mailHost
maxconcurrentconnectionattempts	✓ 32	✓ 32	✓ 32		
maxthreads	✓ 250				
numprocesses	✓ 1				
numthreads : deleted; see maxthreads					
plaintextmncipher		✓ 0	✓ 0		
polldelay	✓ -1				
popbeforesmtpkludgechannel				✓	✓
preauth	✓ 0	✓ 0	✓ 0		✓ 0
preauthtimeout		✓ 600	✓ 600	✓ 600	
preferpoll	✓ 0				
replaypass	✓ 1	✓ 1	✓ 1		
replayformat	✓ %U@%V	✓ %U@%V	✓ %U@%V		✓ %U@%V
requireauthenticationserver		✓ 1	✓ 1		
restrictplainpasswords	✓ 0	✓ 0	✓ 0		✓ 0
searchformat	✓ (uid=%s)	✓ (uid=%s)	✓ (uid=%s)		✓ (uid=%s)
serverdownalert		✓ <i>long string---see text</i>			
servicelist : deleted; see tcp_listen options					
smtpproxypassword				✓	✓
smtprelays				✓	
spoofoftmailbox			✓ 0		

<code>spooftempfail</code>			✓ 0		
<code>spoofmessagefile</code>			✓		
<code>ssladjustciphersuites</code>	✓	✓	✓	✓	✓
<code>sslbacksideport</code>		✓ 0	✓ 0		
<code>sslcachedir</code>	✓	✓	✓	✓	
<code>sslcertnicknames</code> : alias for <code>sslnicknames</code>					
<code>sslcertprefix</code> : deprecated; use <code>base.ssldbprefix</code> instead					
<code>sslenable</code>	✓ 0	✓ 0	✓ 0	✓ 0	
<code>sslkeypasswdfile</code> : deleted					
<code>sslkeyprefix</code> : deprecated; use <code>base.ssldbprefix</code> instead					
<code>sslnicknames</code>	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert
<code>sslports</code> : deleted					
<code>sslsecmodfile</code> : deleted					
<code>storeadmin</code>	✓ complex	✓ complex	✓ complex		✓ complex
<code>storeadminpass</code>	✓	✓	✓		✓
<code>syncldap</code>		✓ 1	✓ 1		
<code>tcpaccess</code>	✓ complex	✓ complex	✓ complex	✓ complex	✓ complex
<code>tcpaccessattr</code>	✓ mailAllowed ServiceAccess	✓ mailAllowed ServiceAccess	✓ mail AllowedService Access		✓ mailAllowed ServiceAccess
<code>timeout</code>	✓ 1800	✓ 1800	✓ 1800	✓ 1800	
<code>usergroupdn</code> : deprecated; see <code>base.ugldapbasedn</code> instead					
<code>use_nslog</code> : deprecated	✓ 1	✓ 1	✓ 1	✓ 1	
<code>usenslog</code> : alias for <code>use_nslog</code>					
<code>virtualdomaindelim</code>	✓ complex	✓ complex	✓ complex		✓ complex
<code>virtualdomainfile</code> : deleted; see <code>vdomain</code> options instead					

Note that many [Base options](#) are relevant to the MMP, including [stressperiod](#) and [stressfdwait](#).

27.1 enable Option Under the MMP

The `enable` MMP option enables the MMP service on `start-msg` startup. The default if this option is not set is 0, but initial configuration may set the option to enable the MMP, as appropriate.

To actually run a proxy server, note that the proxy server must have a [tcp_listen](#) block defined with at least one non-zero port specified within that block; see in particular the [tcp_ports](#) and [ssl_ports](#) `tcp_listen` block options.

27.2 authcachettl Option

The Messaging Server can cache the results of successful LDAP authentication (e.g., when logging into IMAP, POP or SMTP or when the MMP has preauth enabled). `authcachettl` defines the length of time that authentication cache entries are preserved in seconds. Lower values will reduce performance, but result in faster recognition of server password changes. Higher values will increase performance, but result in delayed recognition of server password changes. Changes made to a `userPassword` value in a user's LDAP entry are not seen until the cache entry's time-to-live (TTL) has expired. If you wish to have password changes seen at least every 15 minutes, then set the `authcachettl` value to 900.

27.2.1 Use with base

The `authcachettl` base option specifies the length of time in seconds an authentication cache entry will remain valid. Set to 0 to disable authentication caching.

Note that setting `ldapcachettl` smaller than `authcachettl` causes the entire user entry to expire, thereby also expiring the user authentication information in the user entry.

27.3 authenticationldapattributes Option

The `authenticationldapattributes` Auth option specifies a space-separated list of additional LDAP user attributes to look up and pass to the third-party authentication server. This option is also available at `imapproxy`, `popproxy`, and `vdomain` level (to override, for the respective lookups, the general Auth option). To enable support for a third-party authentication server, set the `authenticationserver` option. For developer instructions and SDK see the directory `msg_svr_base/examples/tpauth`.

27.4 authenticationserver Option

The `authenticationserver` Auth option specifies the hostname and port for a third-party authentication service to use for authentication. The recommended value is `:56` when a third-party authentication service is available on the loopback interface of the server process performing authentication. For developer instructions and SDK see the directory `msg_svr_base/examples/tpauth`.

When not set, the servers will authenticate via LDAP.

27.5 authservice Option

If `authservice` is set to 1 and `authservicettl` is positive, the MMP will allow queries about who is currently logged into the MMP, for the purpose of POP before SMTP relay authentication. This option should almost never be turned on globally; you should configure this by virtual domain. Setting the `authservice` parameter to 1 permits probing of the `authservice` cache with the `xqueryauth ip-address` command over the POP protocol.

27.6 authservicettl Option

The MMP can be configured to remember from which IP address a particular user has authenticated for a period of time. `authservicettl` controls that period of time. This is primarily used for POP before SMTP service, in which case this should be a value greater than 0. A setting of -1 will disable this feature.

27.7 backsideport Option

The `backsideport` option, available for the IMAP Proxy and POP Proxy, specifies the port the MMP will use when connecting to a message store server. This option lets you run a multiplexor and a store server on the same machine, with the store server on a different port. The `smtprelays` option provides equivalent functionality for the SMTP Submit proxy.

27.8 banner Option Under the MMP

The banner MMP option specifies a banner replacement string. The MMP will use the string you specify for its greeting line. The default banner string contains the software name and version information:

Messaging Multiplexor (*product-name version numberbit (built build-date)*)

27.9 banner Option Under the IMAP proxy

The banner IMAP Proxy option specifies a banner replacement string. The IMAP Proxy will use the string you specify for its greeting line. The default banner string contains the software name and version information.

27.10 banner Option Under the POP proxy

The banner POP Proxy option specifies a banner replacement string. The POP Proxy will use the string you specify for its greeting line. The default banner string contains the software name and version information.

27.11 banner Option Under the submit proxy

The banner SUBMIT Proxy option specifies a banner replacement string. The SUBMIT Proxy will use the string you specify for its greeting line. The default banner string contains the software name and version information.

27.12 bethegroup Option

Group ID of for the MMP AService process. DELETED: As of 7u4 (7.4-18.01), the MMP uses the primary group of the user specified by `betheuser`; or in Unified Configuration, the `user` option in `restricted.cnf` is preferred.

27.13 betheuser Option

This specifies the Unix user ID that will be used as the owner of the MMP's AService process (the process group owner will be the primary group of that user). This is deprecated in favor of the `user` option in `restricted.cnf` which will be used preferentially. If this is not set and `restricted.cnf` is not present, the MMP will attempt to use the `imta_user` option from `imta_tailor` instead. For the 7.0.5 release, the MMP will attempt to use `local.serveruid` before checking `imta_tailor`. The value of this option must match the values of the `imta_user` and `local.serveruid` options.

Note that this option is not migrated into the Unified configuration, but is checked to ensure that it is the same as the `user` option in `restricted.cnf`.

27.14 bgmax Option

The `bgmax` option specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See `bgpenalty` for more information.

27.15 bgpenalty Option

When an authentication failure occurs from a particular client IP address, subsequent authentication attempts from that IP address are treated as "BadGuys" and are delayed. If an authentication failure is followed by a successful authentication, the successful authentication is delayed, but the IP address ceases to be treated as a "BadGuy" for subsequent attempts.

`bgpenalty` is the length of time in seconds added to the authentication delay after each failed authentication.

27.16 bgmaxbadness Option

The `bgmaxbadness` option specifies the maximum length of time in seconds for the authentication delay which occurs after a series of failed authentication attempts. See [bgpenalty](#) for more information.

27.17 bgdecay Option

The `bgdecay` option represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See [bgpenalty](#) for more information.

27.18 bglinear Option

The `bglinear` option defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See [bgpenalty](#) for more information.

27.19 bgexcluded Option

The `bgexcluded` option represents a list of excluded IP/mask pairs, or the name of a file to read for these pairs. These client addresses will not be penalized for authentication failure (there is no default value).

27.20 binddn Option

DEPRECATED: Consider using the [ugldapbinddn](#) option instead.

The `binddn` option specifies the Distinguished Name used by the MMP to authenticate to the Directory Server. For schema 1, the `binddn` must have privileges to access the domain tree as specified by the [ldapurl](#) option as well as any users referenced from that domain tree. For schema 2, the `binddn` must have privileges to access the [usergroupdn](#) tree.

The [Messaging Server default directory ACIs](#) require a bind to authenticate users against the Directory Server.

27.21 bindpass Option

Password the MMP uses in conjunction with the `binddn` option. DEPRECATED: Consider using [ugldapbinddn](#) and [ugldapbindcred](#) instead.

27.22 canonicalvirtualdomaindelim Option

The `canonicalvirtualdomaindelim` option (available for `mmp`, `imapproxy`, and `popproxy`) specifies the canonical [virtual domain delimiter](#): that is, the character used by the POP and IMAP proxy to separate the user ID from the appended virtual domain when replaying the user name to the Message Store server. The default is the at-sign character, `@`, so user IDs passed to the Message Store servers have the form `userid@virtual.domain`.

27.23 capability Option

The `capability` IMAP proxy option specifies the capability replacement string. The MMP will use the string you specify for `capability` instead of its default (own) capability to tell IMAP clients what it (or the servers behind it) can do. This variable has no effect in POP3. There is no need to include `STARTTLS` and `AUTH=` extensions as they are added automatically based on the other relevant MMP configuration settings.

There is no need to adjust this string if the backend IMAP servers are entirely from the same version of the Messaging Server installer. Otherwise, it is important to specify a capability list that includes only the features supported by all the backend IMAP servers. The appropriate string can be determined by telnetting to port 143 on each kind of backend server and entering the command `c capability`. Then list only the capabilities present on all backend IMAP servers.

The [ehlokeywords](#) option provides a roughly-equivalent function for the SMTP Submit proxy.

27.24 certmapdn Option

The legacy configuration `certmapdn` has been replaced in Unified Configuration by use of the more general [usergroupdn](#) option.

27.25 certmapfile Option

Legacy config only: The name of the certmap file (for SSL client-cert-based authentications). It may be a full path, but the product's configuration directory will be searched if only a file name is provided.

When this is not set, there is no certmap file and thus the MMP will not request client certificates from the client.

The recommended setting is `certmap.conf`.

This points to the certmap file that will be migrated into the [relevant part of the MMP's](#) Unified configuration.

27.26 clientlookup Option

The `clientlookup` option (available under the `submitproxy` and `vdomain` groups) causes performing a DNS reverse lookup on the client IP address when set to 1. The reverse lookup is performed unconditionally, so the SMTP relay server does not need to perform it. This option may be set on a per hosted domain basis.

Note that a DNS lookup is performed regardless of this setting if hostnames are used in a global `tcpaccess` filter or a per-domain or per-user access filter.

27.27 connecttimeout Option Under the MMP

The `connecttimeout` MMP option specifies how long the MMP should wait for a connection to be established to a back-end mailstore (seconds).

27.28 connecttimeout Option Under the IMAP proxy

The `connecttimeout` IMAP Proxy option specifies how long the MMP IMAP proxy should wait for a connection to be established to a back-end mailstore (seconds).

27.29 connecttimeout Option Under the POP proxy

The `connecttimeout` POP Proxy option specifies how long the MMP POP proxy should wait for a connection to be established to a back-end mailstore (seconds).

27.30 connlimits Option

The `connlimits` option (available under `http`, `imap`, `pop`, `mmp`, `imapproxy`, `popproxy`, `submitproxy`) specifies the maximum number of connections per IP address for the selected server. The syntax is: "realm1,realm2,..." where a realm has the form of address ranges and maximum number of connections expressed as any of the following four forms:

Table 27.2 connlimits Option Value Forms

a.b.c.d e.f.g.h:m	IPv4 address, netmask, connection max
a.b.c.d	IPv4 address
e.f.g.h	network mask
m	maximum connection count
a.b.c.d/p:m	IPv4 address, routing prefix, connection max
a.b.c.d	IPv4 address
p	routing prefix
m	maximum connection count
a.b.c.d e.f.g.h:m	IPv4 address, netmask, connection max
a.b.c.d	IPv4 address
e.f.g.h	network mask
m	maximum connection count
[a/p]:m	IPv6 address, routing prefix, connection max
a	IPv6 address; compressed "::" format allowed

	p	routing prefix
	m	maximum connection count
a.b.c.d e.f.g.h:m		IPv4 address, netmask, connection max
	a.b.c.d	IPv4 address
	e.f.g.h	network mask
	m	maximum connection count
:m		Match any address
	m	maximum connection count

There should be at least one realm of the form ":m" to cover the default case by matching any IPv4 or IPv6 address. To match only IPv4 addresses, use "0.0.0.0/0:m" or "0.0.0.0|0.0.0.0:m". And to match only IPv6 addresses, use "[::0/0]:m".

For backwards compatibility reasons, this option may instead specify the full path to a configuration file name that contains one realm per line. Such a file name must begin with '/'. This usage is deprecated and may be removed in a future release.

27.31 connrejectthreshold Option

The `connrejectthreshold` MMP option specifies the number of connections to accept before rejecting client connections with a soft error at connection time. The default value is computed based on the operating system file descriptor limits for the MMP server process, or 2000 if such file descriptor limits can not be determined. If this is set too high, connections can fail with a 'Too many open files' error.

The default calculation is the file descriptor limit minus 64 (to leave space for log files, LDAP connection pools, internal pipes, etc) divided by 2.

27.32 crams Option

The `crams` option (available under `mmp`, `imaproxy`, `popproxy`, and `vdomain`) is a boolean indicating whether or not to enable Challenge-Response Authentication Mechanisms (CRAMs) including APOP and CRAM-MD5. For this to work, passwords must be stored in LDAP in plain text format and the `bindDN` must have read access to the `userPassword` attribute -- or in more modern configurations `ugldapbinddn` must have read access to `ugldapbindcred`. If `crams` is not set, the `has_plain_passwords` option will be used instead.

27.33 debugkeys Option

The `debugkeys` option (available under `base`, `mmp`, `imaproxy`, `popproxy`, `submitproxy`, and `vdomain`) specifies a space-separated list of keywords used to enable various optional debugging facilities. Currently recognized keywords are listed in the table below.

Table 27.3 Keywords That May Be Included in debugkeys Option Value

Keyword	Function
<code>admindebug</code>	log msadmin basic diagnostics (new in 7.0.5)
<code>adminerr</code>	log msadmin http errors (new in 7.0.5)

adminlog	log msadmin http connections (new in 7.0.5)
adminrecv	log msadmin request information (new in 7.0.5)
adminverb	log msadmin verbose information (new in 7.0.5)
adminxmit	log msadmin transmission information (new in 7.0.5)
archive	log diagnostics for imapd archiving interface
authserv	log auth server protocol communications (new in 7.0.5)
bind	log additional details about some TCP socket bind attempts
certmap	log debug-level details about certificate mapping operations used for client certificate authentication (new in 7.0.5)
connect	log additional details about some TCP connection attempts (more coverage in 7.0.5)
enssub	enable logging of ENS subscribe/unsubscribe events at notice level (new in 8.0)
gdisp	help diagnose generic dispatcher API issues
gdwork	GDisp worker thread information
gdcvar	GDisp condition variables (not presently used by the MMP).
hula	log state changes in HULA (user lookup / authentication, new in 7.0.5)
maparse	Diagnostics for IMAP mail access parser (new in 8.0). The set of IMAP commands this covers presently includes APPEND, STORE, SETMETADATA, SEARCH, ESEARCH, SORT, THREAD. Additional commands will be added over time. This is refreshable.
metermaid	log transcript of metermaid client used to limit IMAP password expiration alerts (new in 7.0.5)
perf	log performance-related timestamps particularly with respect to MMP authentication
ldap	log a directory protocol trace
lpool	log ldap connection activity (mostly INFO & DEBUG level, new in 7.0.5); for MTA output, see also the os_debug MTA option
search	log IMAP search and sort command processing at DEBUG level (new in 7.0.5)
tls	enable additional SSL/TLS debugging (presently just lists active cipher suites in the MMP log)

27.33.1 Use with base

The `debugkeys` base option specifies a space-separated list of keywords used to enable various optional debugging facilities.

27.34 defaultdomain Option

When POP, IMAP and SMTP users authenticate, they typically provide an unqualified user ID (a user ID without a domain portion). The value of the `defaultdomain` option is appended to unqualified user IDs. When used as an MMP virtual domain option, this allows a single MMP server with multiple IP addresses to support unqualified user IDs for multiple hosted domains. This may also be set as a service-wide option.

27.34.1 Use with base

The `defaultdomain` base option specifies the Messaging Server default domain. This is used to determine whether a domain is the default domain or a hosted domain.

Normally the `defaultdomain` base option is set to an appropriate value during initial configuration.

The MTA has a "twin" option, `ldap_default_domain`, that can override the `defaultdomain` base option for MTA purposes. See the description of `ldap_default_domain` for details on how the MTA uses the `defaultdomain` value (if `ldap_default_domain` is not set).

27.35 domainallowed Option

The `domainallowed` option specifies [access filters](#) specifying which domains and/or IP addresses are allowed access for the selected server.

27.35.1 Use with http

The `domainallowed` MSHTTP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed HTTP access.

27.35.2 Use with imap

The `domainallowed` IMAP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed IMAP access.

27.35.3 Use with pop

The `domainallowed` POP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed POP access.

27.35.4 Use with ens

The `domainallowed` ENS option specifies [access filters](#) specifying which domains and/or IP addresses are allowed ENS access.

27.35.5 Use with eval_ldapd

The `domainallowed` option under `eval_ldapd` allows setting [access filters](#) specifying which domains and/or IP addresses are allowed evaluation ldapd access.

27.36 domainnotallowed Option

The `domainnotallowed` option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed access for the selected server.

27.36.1 Use with eval_ldapd

The `domainnotallowed` option under `eval_ldapd` allows setting [access filters](#) specifying which domains and/or IP addresses are *not* allowed evaluation ldapd access.

27.36.2 Use with http

The `domainnotallowed MSHTTP` option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed HTTP access.

27.36.3 Use with imap

The `domainnotallowed IMAP` option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed IMAP access.

27.36.4 Use with pop

The `domainnotallowed POP` option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed POP access.

27.36.5 Use with ens

The `domainnotallowed ENS` option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed ENS access.

27.37 domainsearchformat Option

The `domainsearchformat` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) specifies a printf-style format string with which to construct Users/Groups LDAP queries for the user's mailhost when [hosteddomains](#) is enabled. If `domainsearchformat` is not set, then the [searchfilter](#) option will be used (regardless of whether `hosteddomains` is set). Valid escape sequences are:

```
%s (userid+virtualdomain)
%U (userid only)
%V (virtual domain only)
%C (client IP address)
%S (server IP address)
%D (client cert subject DN)
%o (original user as passed from client)
```

27.38 ehlokeywords Option

The `ehlokeywords` option under `submitproxy` or `vdomain` specifies a list of EHLO extension keywords for the proxy to pass through to the client, in addition to the default set. The MMP normally removes any unrecognized EHLO keywords from the EHLO list returned by an SMTP relay, but `ehlokeywords` specifies additional EHLO keywords which should *not* be removed from the list. The default is empty, but the SMTP Submit proxy supports the following keywords (there is no need to list them in this option): 8BITMIME, PIPELINING, ENHANCEDSTATUSCODES, EXPN, HELP, ETRN, SIZE, STARTTLS, AUTH.

27.39 failovertimeout Option

The `failovertimeout` option under `submitproxy` or `vdomain` specifies how many seconds the MMP will wait for an SMTP server connection prior to failing over to the next SMTP server in the list. If a connection to an SMTP relay fails, the MMP avoids trying that relay for a number of minutes equivalent to the failover time-out. For example, if the failover time-out is 10 seconds, and a relay fails, the MMP does not try that relay again for 10 minutes.

27.40 hostedomains Option

The `hostedomains` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) specifies whether the MMP should use Hosted Domains. The default is 1, meaning that Hosted Domains are supported. If `hostedomains` is set to 0, then the value of the [searchfordomain](#) authentication option controls the behavior during user authentication lookups.

If you are using the Messaging Server directory schema (LDAP Schema, v1 or LDAP Schema, v2), `hostedomains` should be set to the default value of 1.

If set to 0, then the MMP assumes the server supports only one domain and the [ugldapbasedn](#) option points to a directory subtree containing all users supported by the server, each user with a unique UID. Setting `hostedomains` to "0" is not recommended as even a small company is likely to eventually go through a name change or acquisition where support for multiple domains would be helpful.

When set to 1, the MMP honors the following additional options (for legacy configuration, these appear in the MTA's `option.dat` configuration file):

```
ldap_schemalevel
ldap_domain_filter_schema1
ldap_domain_filter_schema2
ldap_attr_domain1_schema2
ldap_attr_domain2_schema2
ldap_global_config_templates
ldap_attr_domain_search_filter
ldap_domain_attr_basedn
ldap_domain_attr_canonical
ldap_domain_attr_alias
```

These settings may be used to enable LDAP Schema, v2 with the MMP.

27.41 ipv6in Option

When set to a value of 1, the `ipv6in` option instructs Messaging Server to accept inbound IPv6 connections for all services provided that the host has at least one network interface configured for IPv6. Services specifically configured to listen on only IPv4 interfaces cannot also accept inbound IPv6 connections. When set to a value of 0, inbound IPv6 connections are not allowed.

Inbound IPv4 connections will always be permitted.

27.42 ipv6out Option

When set to a value of 1, the `ipv6out` option instructs Messaging Server to attempt outbound IPv6 connections for all services provided that the host has at least one network interface configured for IPv6. Services specifically configured to bind their source IP address only to IPv4 interfaces cannot attempt IPv6 outbound connections. For example, an SMTP client bound to a specific IPv4 interface cannot then establish an outbound IPv6 connection. When set to a value of 0, outbound IPv6 connections are not allowed.

When set to a value of 1, outbound services will attempt DNS lookups of both A and AAAA records. Connection attempts will then be made in the order dictated by the `ipv6sortorder` option. Note the DNS lookups will always request A records. This option only controls whether or not AAAA records are also requested.

27.43 langlist Option

The `langlist` option (under `mmp` or `imapproxy`) controls the list of supported languages returned by the IMAP LANGUAGE extension when issued to the MMP prior to authentication.

27.44 ldapcachesize Option

The MMP can cache results of user searches. The `ldapcachesize` option (available under `mmp`, `imapproxy`, `popproxy`, `submitproxy`, and `vdomain`) defines the number of cache entries; `ldapcachettl` defines the length of time the entries are preserved in seconds. Lower values will reduce performance, but result in faster recognition of LDAP user entry changes. Higher values will increase performance, but result in delayed recognition of LDAP user entry changes. If this is not set, then the `authcachesize` option's value will be used instead. If `ldapcachesize` is set, it will override `authcachesize` for MMP purposes only.

27.45 ldapcachettl Option

The MMP can cache results of user searches. The `ldapcachesize` option defines the number of cache entries; `ldapcachettl` defines the length of time the entries are preserved in seconds. Lower values will reduce performance, but result in faster recognition of LDAP user entry changes. Higher values will increase performance, but result in delayed recognition of LDAP user entry changes.

Note that setting `ldapcachettl` smaller than `authcachettl` causes the entire user entry to expire, thereby also expiring the user authentication information in that user entry.

27.46 ldappendingoplimit Option

The `ldappendingoplimit` option (available under `mmp`, `imapproxy`, and `popproxy`) specifies the number of in-progress LDAP connections the MMP will allow before it will delay incoming connections to wait for previous LDAP operations to complete. This prevents a denial-of-service attack on the MMP from impacting the LDAP server.

The default has changed from 20 to 128 in the 7.0.5 release.

27.47 ldaprefreshinterval Option

The `ldaprefreshinterval` option (available under `mmp`, `imapproxy`, and `popproxy`) specifies the seconds that the MMP will keep a connection open to the LDAP server. When the MMP notices that the refresh interval has passed, the MMP will close the LDAP connection and open a new one.

27.48 ldaptimeout Option

The `ldaptimeout` option specifies the number of seconds the MMP will wait for an LDAP operation to complete before it will attempt a failover to a backup LDAP server or fail the operation. DEPRECATED: Consider using the `ldapsearchtimeout` and `ldapmodifytimeout` options instead.

27.49 ldapurl Option

DEPRECATED as of 7.0.5: consider using `ugldaphost`, `ugldapussl`, `dcroot` and `ugldapbasedn` instead.

The `ldapurl` option, available for the MMP, IMAP Proxy, POP Proxy, and SUBMIT Proxy, specifies an LDAP URL pointing to the top of the site's DC directory tree (used by schema 1), if `hosteddomains` is set to `yes` (default). If `hosteddomains` is set to `no`, then `ldapurl` points to a directory subtree containing all users supported by the server. Prior to 7.0.5, this option was needed for the MMP to operate correctly. For schema 2 support, the `usergroupdn` option must be set and is used instead of the path portion of this URL.

SSL (LDAPS) is supported, but the SSL configuration must also be correct, and SSL-enabled.

To enable failover, the host part of the URL may be a space-separated list of hosts. Be sure to enclose the entire URL in double-quotes if it contains a space. For example:

```
"ldap://ldap1.example.com ldap2.example.com/o=internet"
```

27.50 logdir Option

The `logdir logfile` option, `component.logfile.logdir`, specifies the directory path for log files. If this is not specified, log files will be placed in the `msg-install-path/data/log` directory.

For the MTA, the `mta.logfile.logdir` option is only used by Message Store insertion tasks. It specifies the directory path to the `imta` log file used for Message Store insertion (`ims_master`, `LMTP`). It is not used by other parts of the MTA which always log to the default location. The default location is `msg-install-path/data/log`. Changing that path to a soft-link is supported.

27.51 loglevel Option Under the MMP

The MMP's `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than 'error'. For backwards compatibility, MMP configuration files may use integer settings from 3 to 7 for 'error' to 'debug' respectively, or 0 for `nolog`.

27.52 loglevel Option Under the IMAP proxy

The IMAP Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than `error`.

27.53 loglevel Option Under the POP proxy

The POP Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than `error`.

27.54 loglevel Option Under the submit proxy

The SUBMIT Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than `error`.

27.55 mailhostattrs Option

The `mailhostattrs` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) specifies a space-separated list of LDAP attributes identifying the user's mail host; the default is simply `mailHost`. The multiplexor tries each attribute returned by the search in the order specified by the list to identify the mail store where that user's mail lives.

This is rather analogous to the MTA's `ldap_mailhost` option.

27.56 maxconcurrentconnectionattempts Option

The `maxconcurrentconnectionattempts` option (available under `mmp`, `imapproxy`, and `popproxy`) specifies the number of outstanding connection attempts permitted to the same backend mailstore. If this is exceeded, users on that mailstore will have their connections rejected with a temporary service outage error. This limit prevents a DNS or mailstore outage of one server from consuming all the MMP worker threads.

The default has changed from 10 to 32 in the Messaging Server 7.0.5 release.

27.57 maxthreads Option Under the MMP

The `maxthreads` MMP option specifies the maximum number of threads allowed per server process for the selected server. The MMP does not count worker threads attempting to lookup or connect to a back-end server against this limit; see the separate `maxconcurrentconnectionattempts` option to limit such connections.

27.58 numprocesses Option Under the MMP

The `numprocesses` MMP option specifies the number of MMP AService processes. Note that the Watcher must be enabled for `stop-msg` to correctly shut down all processes if this is set to a value larger than one.

27.59 numthreads Option

The `numthreads` MMP option had specified the maximum number of worker threads to permit for the MMP.

DELETED: This MMP AService.cfg option was removed in favor of the `maxthreads` option.

27.60 plaintextmncipher Option

If the `plaintextmncipher` option is > 0 for a service, then disable use of plaintext passwords over that service unless a security layer (SSL or TLS) is activated for the selected service. This forces users to enable SSL or TLS on their client to login which prevents exposure of their passwords on the network.

27.60.1 Use with http

If the `http.plaintextmncipher` option 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.

27.60.2 Use with imap

If the `imap.plaintextmncipher` option 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.

27.60.3 Use with pop

If the `pop.plaintextmncipher` option 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.

27.60.4 Use with mta

If the `plaintextmncipher` MTA option 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. This option in the `mta` group presently also applies to the MTQP and ManageSieve servers.

27.61 polldelay Option

Solaris-only. The `polldelay` (IMAP and MMP) option specifies the wait time before calling `poll()` in milliseconds. Workaround for poll performance bug on Solaris (6438988, 6379476). Setting this to `-1` activates a different workaround as of 7 update 4 patch 24. The alternate code tries to keep the size of the poll array relatively constant and instead uses `-1` in the poll array

for inactive descriptors. The poll array will be larger, but change size less frequently. To date this appears to noticeably improve performance under stress.

The default has changed from 1 to -1 in the Messaging Server 7.0.5 release. In addition, poll is no longer used in the Messaging Server 7.0.5 release (and thus this option is ignored) unless [preferpoll](#) is set.

27.62 popbeforesmtpkludgchannel Option

The `popbeforesmtpkludgchannel` SUBMIT Proxy or Virtual Domain option specifies 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](#).

27.63 preauth Option

The `preauth` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) enables pre-authentication by the MMP. When `preauth` is set to 1 (`yes` in legacy configuration), a user is authenticated against the LDAP server before a connection is made to the backend mailstore server. When `preauth` is set to 0 (`no` in legacy configuration), the MMP connects to the backend mailstore server and simply replays the authentication information. Because of the additional authentication step, `preauth` reduces the overall performance, but protects the backend mailstore servers from denial-of-service attacks by unapproved users. `preauth` is mandatory for the POP-before-SMTP feature of the MMP.

When using [hosteddomains](#), the `mailAccessProxyPreAuth` attribute in the domain node in the LDAP server overrides this option.

27.64 preauthtimeout Option

The `preauthtimeout` option (available for the IMAP Proxy, POP Proxy, and SMTP SUBMIT Proxy) specifies the MMP session timeout prior to authentication.

27.65 preferpoll Option

To improve performance, the IMAP and MMP servers use Solaris Event Completion Ports on Solaris instead of the poll system call starting with the Messaging Server 7.0.5 release. Since the Messaging Server 8.0.1 release, the servers use `epoll` on Linux instead of the poll system call. Setting the `preferpoll` option (available at base and MMP level) to 1 will revert to use of the standard Posix poll API instead. When `preferpoll` is set to 1, then the [polldelay](#) option also applies.

27.66 replayformat Option

The `replayformat` option takes an argument of a printf-style format string that says how to construct the user ID for replay to the Message Store server. Valid escape sequences are:

```
%s (user@domain where '@' is the canonical domain delimiter)
%o (original user as sent by the client)
%U (userid only)
```

```
%V (virtual domain only)
%A[attr] (value of user's attribute "attr")
```

For example, %A[uid]@%V for a user with joe as the user ID and domain=siroe.com would yield:

```
joe@siroe.com
```

For the MMP, when using [hosteddomains](#), the `mailAccessProxyReplay` attribute in the domain node in the LDAP server overrides this option.

27.66.1 Use with http

The `replayformat` MSHTTPD option, `http.replayformat`, specifies the format for authentication replay from `mshttpd` to IMAP backend. Supports %o for original userid, %s for user@domain, %U for userid only, %V for virtual domain, %A[attr] for value of specified user's attribute.

27.67 replaypass Option

The `replaypass` option (available under `mmp`, `imapproxy`, and `popproxy`) is a boolean indicating whether to replay the end-user's password to the back-end IMAP or POP server. If this is set to 0, then the password is not replayed and administrative proxy authentication is used, so the [storeadminpass](#) option must also be set.

27.68 requireauthenticationserver Option

When an authentication server is configured using the [authenticationserver](#) option, and `requireauthenticationserver` is 1 (the default), that server must be running and responding to requests or authentication will not succeed. If `requireauthenticationserver` is set to 0, then built-in authentication mechanisms will be permitted even if the authentication server ceases to respond to requests.

27.69 restrictplaintextpasswords Option

When the `restrictplaintextpasswords` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) is set to 1, this will forbid use of plaintext passwords unless an SSL/TLS security layer is active. If this is not set, the [plaintextmincipher](#) option will be used.

27.70 searchformat Option

The `searchformat` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) specifies a printf-style format string with which to construct Users/Groups LDAP queries to locate a user in the directory (and in particular to determine the user's mailhost) when [hosteddomains](#) is disabled -- or if [hosteddomains](#) is enabled but `domainsearchformat` is not set. Valid escape sequences are:

```
%s (userid+virtualdomain)
%U (userid only)
```

%V (virtual domain only)
%C (client IP address)
%S (server IP address)
%D (client cert subject DN)
%o (original user as passed from client)

27.71 serverdownalert Option

The `serverdownalert` IMAP Proxy option specifies the string returned to client in an IMAP ALERT message when the MMP cannot connect to a user's store server.

27.72 servicelist Option

For legacy MMP config only: Specifies which services to start and the ports/interfaces on which the MMP will listen for those services. Services are listed all on a single line in the following format: `SERVICENAME @ HOSTPORT [| HOSTPORT]`

Where *SERVICENAME* is `popproxyaservice`, `imapproxyaservice` or `smtpproxyaservice` (any prefix to these names is ignored for backwards compatibility).

This option must be set for the MMP to function correctly. The initial configuration (as of release 7 Update 3) will set this to:

```
ImapProxyAService@143 PopProxyAService@110
```

Note that this option is not migrated directly to the Unified configuration. Equivalent settings appear in the [tcp_listen](#) sections of the `popproxy`, `imapproxy` and `submitproxy` sections.

27.73 smtpproxyaservice Option

The `smtpproxyaservice` option specifies the password the MMP uses to authorize source channel changes on the SMTP relay servers. This option is available under `mta`, `submitproxy`, and `vdomain`. To use this functionality, that is, to use the MMP's SMTP SUBMIT Proxy, the option must be set under `mta` on the MTA back end, and must be set for the MMP's SMTP SUBMIT Proxy (thus under either `submitproxy` if being set in general, or under `vdomain` if it is only to be applied for a particular virtual domain) on a front end MMP system, and **these values must match** between front and back ends! The option has no default.

If the `mta.smtpproxyaservice` option is not set, client attempts to use the XPEHLO command will receive an error (issued by the MTA SMTP server):

```
503 5.5.0 Proxy support is not enabled.
```

If `mta.smtpproxyaservice` is set but its value does not match the MMP's value, client attempts to use the XPEHLO command will receive an error (issued by the MTA's SMTP server):

```
535 5.7.8 SMTP proxy authentication check failed.
```

Note that the legacy configuration equivalent of the Unified Configuration `mta.smtpproxyaservice` option was the [PROXY_PASSWORD](#) TCP/IP-channel-specific

option (which in legacy configuration, was set in the SMTP server's TCP/IP channel option file). (The MTA option `smtpproxypassword` was introduced in MS 7.0u5.) And the legacy configuration equivalent of the Unified Configuration `submitproxy.smtpproxypassword` option (as well as the `vdomain.smtpproxypassword` option) was set as `SmtProxyPassword` in the `SmtProxyAService.cfg` file.

27.74 smtprelays Option

The `smtprelays` SUBMIT Proxy option specifies a space-separated list of SMTP SUBMIT server hostnames (with optional port) to use for round-robin relay. These SMTP servers must support the [XPEHLO extension](#). Setting the `smtprelays` option is mandatory in order to use the SUBMIT Proxy; it has no default. For example:

```
sesta.example.com:485 gonzo.example.com mothra.example.com
```

27.75 spoofemptymailbox Option

If the `spoofemptymailbox` POP Proxy option is set to 1 (default is 0) and the user's POP server is unavailable, the MMP will simply return an empty mailbox listing. Turning this option on will override the `spoofmessagefile` option. We have received reports that this will cause certain clients (including Microsoft® Outlook) to re-download the mailbox when the back-end server comes back online.

27.76 spooftempfail Option

If the `spooftempfail` POP Proxy option is set to 1 (default is 0) and a temporary authentication error occurs subsequent to locating the user in LDAP, the MMP will simply return an empty mailbox listing or if `spoofemptymailbox` is 0 and `spoofmessagefile` is set, then the spoof message file will be used. We have received reports that this will cause certain clients (including Microsoft Outlook) to re-download the mailbox when the temporary condition is resolved.

A temporary authentication error can occur as a result of `defer` (or as of Messaging Server 7.5 `defer-submit`) or hold mail user status (`mailUserStatus`) or as a result of hold mail domain status (`mailDomainStatus`) prior to connecting to the back-end POP server, and can also occur if the back-end server returns a `[SYS/TEMP]` authentication failure. In the former case, the MMP's LDAP cache settings apply.

27.77 spoofmessagefile Option

The `spoofmessagefile` POP Proxy option specifies the file to use for POP3 inbox spoofing. The MMP can imitate a base-functionality POP3 server in case it can't connect to a client's store machine. In such a situation, the MMP creates an inbox for the user and places this one message into it. The format of the message contained in this file should conform to dot-stuffed [RFC 822](#) (including the final '.').

By default, there is no spoof message file.

27.78 ssladjustciphersuites Option

The `ssladjustciphersuites` option allows adjusting which SSL cipher suites are enabled or disabled. SSL cipher suites control the level of protection required between SSL client and server. Different cipher suites have different properties and use different cryptographic algorithms. At any time a specific cryptographic algorithm might be weakened or compromised by new research in cryptography. The ability to change the default cipher suites allows the software to adapt as security technology changes. In addition as CPUs get faster, the key size necessary to provide several years of comfortable protection increases, even if the algorithm is considered state-of-the-art.

The default set of SSL cipher suites used may change over time as more secure ones are introduced and weaker ones are deprecated. It is expected most deployments will be happy with the default set of cipher suites and it is generally not a good idea to adjust the available cipher suites without reason. However, here are some scenarios where it may be helpful to adjust cipher suites:

1. a site with specific security policies may wish to provide a fixed list of cipher suites to use that is set by site policy rather than simply using state-of-the-art suites provided by the NSS library. Such a site would typically configure this setting to '-ALL,...' where '...' contains the cipher suite names.
2. A site which is experimenting with higher performance or more secure cipher suites that require installation of special server certificate types, for example the elliptic curve cipher suites. Such a site would enable these additional suites once installation was complete using a setting such as '+TLS_ECDH_RSA_WITH_AES_128_CBC_SHA' to enable an ECDH_RSA cipher suite from [RFC 4492](#).
3. If a site is forced to continue supporting a particularly old client that only supports weak cipher suites, they can be explicitly enabled (for example 'WEAK+DES' enables the single-DES cipher suites).
4. In the event the cryptographic research community discovers a vulnerability in one or more of the ciphers enabled by default, this provides a mechanism to immediately disable those ciphers. For example, to disable all ciphers using the 'RC4' algorithm, simply set '-RC4'.

As of 2008-Jan-29, the available cipher suites in the NSS library are as follows:

```
TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_AES_256_CBC_SHA,  
TLS_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_RC4_128_SHA,  
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,  
SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,  
TLS_RSA_WITH_AES_128_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,  
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA,  
SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA,  
SSL_DHE_DSS_WITH_DES_CBC_SHA, SSL_RSA_FIPS_WITH_DES_CBC_SHA,  
SSL_RSA_WITH_DES_CBC_SHA, TLS_RSA_EXPORT1024_WITH_RC4_56_SHA,  
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA, SSL_RSA_EXPORT_WITH_RC4_40_MD5,  
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5, SSL_RSA_WITH_NULL_SHA,  
SSL_RSA_WITH_NULL_MD5, TLS_ECDH_ECDSA_WITH_NULL_SHA,  
TLS_ECDH_ECDSA_WITH_RC4_128_SHA,  
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,  
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,  
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,  
TLS_ECDHE_ECDSA_WITH_NULL_SHA, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA,  
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,  
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
```

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDH_RSA_WITH_NULL_SHA, TLS_ECDH_RSA_WITH_RC4_128_SHA, TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_NULL_SHA, TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA. This list excludes the SSL2 cipher suites as Messaging Server has not supported SSL2 since the 6.0 release. While the standard names for cipher suites (as published in TLS RFCs) are preferred, there is limited support for legacy names used in previous releases and for some OpenSSL names. Note that the TLS_DHE_* cipher suites are only available for outgoing connections from Messaging Server.

Starting with version 7.0.5.31.0 (NSS 3.16), the following additional cipher suites are available when the [tlsv12enable](#) option is set:

TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_NULL_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.

27.79 sslbacksideport Option

The `sslbacksideport` IMAP Proxy and POP proxy option specifies the port number to which the MMP will try to connect on the store servers using SSL if an SSL connection was made to the MMP. If this parameter is not set, the MMP will not use SSL when connecting to the store. There are no default values, but ports 993 and 995 are recommended for IMAP and POP, respectively. (On the relevant back end Message Store systems, see the [sslport](#) option for IMAP and POP, respectively, to see on what ports the backend Message Store is actually listening for such incoming SSL connections.)

27.80 sslcachedir Option

The `sslcachedir` option specifies the SSL session cache directory used to track SSL sessions across multiple connections by the MMP. Prior to 7.0.5.31.0, this also controlled the location of the SSL database files and defaulted to the config directory. As of the 7.0.5.31.0 release, the [ssldbpath](#) base option takes precedence over this option for specifying the location of SSL database files.

NOTE: In order for results to be predictable, this option must be the same for the IMAP, POP, and SMTP proxies -- that is, in legacy configuration it must be the same in the files `ImapProxyAService.cfg`, `PopProxyAService.cfg` and `SmtpproxyAService.cfg`, or in Unified Configuration any settings of this option for the various proxies must match.

27.81 sslcertprefix Option

Filename prefix to the SSL certificate database file. The certificate database file must be in the directory specified by the [ssldbpath](#) setting. No prefix will be used by default. DEPRECATED: The [ssldbprefix](#) option should be used instead.

NOTE: In order for results to be predictable, this option must be the same for the IMAP, POP, and SMTP proxies -- that is, in legacy configuration it must be the same in the files `ImapProxyAService.cfg`, `PopProxyAService.cfg` and `SmtpproxyAService.cfg`, or in Unified Configuration any settings of this option for the various proxies must match.

27.82 sslkeypasswdfile Option

File location for the passwords that protect access to the private key file. Passwords may be null if the key is not password-protected. This option is deprecated, and will go away. Use is not recommended as all product components except the MMP: simply use the default `sslpassword.conf` name. If multiple SSL configurations are required, use [sslpath](#) to relocate these files instead.

NOTE: In order for results to be predictable, this option must be the same for the IMAP, POP, and SMTP proxies -- that is, in legacy configuration it must be the same in the files `ImapProxyAService.cfg`, `PopProxyAService.cfg` and `SmtpproxyAService.cfg`, or in Unified Configuration any settings of this option for the various proxies must match.

27.83 sslkeyprefix Option

Filename prefix to the SSL key database file. The key database file must be in the directory specified by the [sslpath](#) setting. No prefix will be used by default. DEPRECATED: The [sslprefix](#) option should be used instead.

NOTE: In order for results to be predictable, this option must be the same for the IMAP, POP, and SMTP proxies -- that is, in legacy configuration it must be the same in the files `ImapProxyAService.cfg`, `PopProxyAService.cfg` and `SmtpproxyAService.cfg`, or in Unified Configuration any settings of this option for the various proxies must match.

27.84 sslnicknames Option

The `sslnicknames` option (available at base level, and also available at a variety of more specific levels) specifies a list of the nicknames of the certificates in the SSL certificate database to offer as the server certificate. Only one nickname of each certificate type is permitted (*e.g.*, one RSA certificate, one DSS certificate) so normally only one will be specified.

27.84.1 Use with base

List of SSL/TLS server certificate nicknames (only one per certificate type) to offer clients if SSL/TLS enabled. This base level list may be overridden for particular services.

27.84.2 Use with http

The `sslnicknames MSHTTP` option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for HTTP to offer clients if SSL/TLS enabled. Overrides for HTTP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

27.84.3 Use with imap

The `sslnicknames` IMAP option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for IMAP to offer clients if SSL/TLS enabled. Overrides for IMAP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

27.84.4 Use with mta

The `sslnicknames` MTA option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for MTA to offer clients if TLS is enabled. Overrides for the MTA the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter). This option in the `mta` group presently also applies to the MTQP and ManageSieve servers.

27.84.5 Use with pop

The `sslnicknames` POP option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for POP to offer clients if SSL/TLS enabled. Overrides for POP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

27.84.6 Use with ens

The `sslnicknames` ENS option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for HTTP to offer clients if SSL/TLS enabled. Overrides for HTTP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

27.85 `sslsecmodfile` Option

Security module database file name. If you have hardware accelerators for SSL ciphers, this file describes them to the Messaging Server. DELETED: Support removed in 7.0.5 as customized module file name feature in NSS goes away with new `cert9.db/key4.db/pkcs11.txt` format.

27.86 `storeadmin` Option

`storeadmin` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) is set to the user name of the store administrator for proxy authentication and is necessary to support SSL client certificates and RFC 2595-style proxy authentication. If this is not set and the `admins` Message Store option is single-valued, the value of the `admins` option will be used instead. Otherwise MMP features requiring store admin credentials will be disabled.

27.87 `storeadminpass` Option

The `storeadminpass` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) specifies the password for the `store administrator` used by MMP proxy authentication necessary to support SSL client certificates and RFC 2595-style proxy authentication. There is no default for `storeadminpass`.

27.88 `syncldap` Option

The `syncldap` option (available under `imapproxy` and `popproxy`) causes the IMAP Proxy or POP Proxy to do synchronous LDAP lookups instead of async LDAP lookups. This may improve performance under high load conditions by reducing the number of round-trips through the main poll dispatch loop. Make sure the `ldappendingoplimit` option is set to a value below `maxthreads` if this is enabled. Set this to 0 to get the pre-7.0.5 async LDAP behavior. The async LDAP behavior is deprecated and may be removed in a future release.

27.89 tcpaccess Option

The `tcpaccess` option (available under `mmp`, `imapproxy`, `popproxy`, `submitproxy`, and `vdomain`) specifies wrap-style filters that describes TCP access control for the MMP (globally). If this is not set, then the `domainallowed` and `domainnotallowed` options for the service will be used instead.

See "Configuring Client Access to POP, IMAP, and HTTP Services" in the "Configuring Security and Access Control" chapter of the Messaging Server Administrator's Guide for background on [filter syntax](#).

27.90 tcpaccessattr Option

The `tcpaccessattr` option -- available for the MMP, IMAP proxy, POP proxy, and virtual domains -- specifies the name of a per-user LDAP attribute that contains a [wrap-style filter](#) describing the [TCP access](#) control for the user.

27.91 timeout Option Under the MMP

The `timeout` MMP option specifies the session timeout in seconds. To be standards-compliant, the value of this option must not be set lower than 1800 seconds (30 minutes) for IMAP, 600 seconds (10 minutes) for POP or SMTP.

27.92 timeout Option Under the IMAP proxy

The `timeout` IMAP Proxy option specifies the session timeout in seconds. To be standards-compliant, the value of this option must not be set lower than 1800 seconds (30 minutes) for IMAP.

27.93 timeout Option Under the POP proxy

The `timeout` POP Proxy option specifies the session timeout in seconds. To be standards-compliant, the value of this option must not be set lower than 600 seconds (10 minutes) for POP.

27.94 timeout Option Under the submit proxy

The `timeout` SUBMIT Proxy option specifies the session timeout in seconds. To be standards-compliant, the value of this option must not be set lower than 600 seconds (10 minutes) for SMTP.

27.95 use_nslog Option

The `use_nslog` option may be set to 1 to enable use of `nslog()` for debugging output. This then enables the use of the [logfile options](#), `component.logfile.option-name`, (or `logfile.component.*` in legacy configuration) for controlling logfile creation and rollover. Note that `loglevel` is not supported for the [dispatcher](#) and [job_controller](#) as their logging level is still controlled by use of their respective [debug](#) options. `use_nslog` defaults to 0 for the dispatcher and `job_controller` and 1 for the MMP.

27.95.1 Use with dispatcher

The `use_nslog` Dispatcher option may be set to 1 to enable use of `nslog()` for Dispatcher debug log files. The default is 0. When `use_nslog` has been enabled, see also the [logfile options](#) that may be set as `dispatcher.logfile.*`. Note that the `loglevel` option is *not* supported for the [Dispatcher](#) as its debug level is controlled instead by use of the [debug](#) Dispatcher option, `dispatcher.debug`.

27.95.2 Use with job controller

The `use_nslog` Job Controller option may be set to 1 to enable use of `nslog()` for Job Controller debug log files. The default is 0. When `use_nslog` has been enabled, see also the [logfile options](#) that may be set as `job_controller.logfile.*`. Note that the `loglevel` option is *not* supported for the [Job Controller](#) as its debug level is controlled instead by use of the [debug](#) Job Controller option, `job_controller.debug`.

27.96 useNSLog Option

The legacy configuration `useNSLog` has been replaced in Unified Configuration by [use_nslog](#).

27.97 usergroupdn Option

The `usergroupdn` option (available for the MMP, IMAP Proxy, POP Proxy, and Submit Proxy) specifies the baseDN for user, group and domain searches by the MMP in LDAP Schema, v2 mode. It is also used for client certificate mapping lookups in LDAP Schema, v1 mode. If this is not set, it defaults to the value of the [ugldabasedn](#) option.

27.98 virtualdomaindelim Option

The `virtualdomaindelim` option (available under `mmp`, `popproxy`, `imapproxy`, and `vdomain`) takes a string value specifying the acceptable virtual domain delimiters. Any character in this string will be treated as a domain delimiter in a user ID received by the MMP. (The MMP searches user IDs from the end.) If this is not set, then the [loginseparator](#) option's value will be used instead.

27.99 virtualdomainfile Option

For legacy MMP config only: The name of the file containing your virtual domain mapping (a full path may be provided, but the product's configuration directory is used if no path is provided).

The recommended setting is `vdmap.cfg`. Uncomment this line in the configuration file to enable support for virtual domains.

This option is deleted during Unified configuration migration and its contents are included in `<vdomain>` option groups.

Chapter 28 tcp_listen options

28.1 tcp_ports Option Under tcp_listen	28-1
28.2 ssl_ports Option Under tcp_listen	28-1
28.3 listen_addresses Option Under tcp_listen	28-1
28.4 backlog Option Under tcp_listen	28-1

Named `tcp_listen` groups can appear under the `imapproxy`, `popproxy`, and `submitproxy` groups. Several options may be set under `tcp_listen`.

28.1 tcp_ports Option Under tcp_listen

Under a top level `imapproxy`, `popproxy`, or `submitproxy` component, inside a named `tcp_listen` group, the `tcp_ports` option specifies the TCP port(s) on which that proxy server listens; *e.g.*:

```
msconfig> set imapproxy.tcp_listen:imap-proxy-1.tcp_ports 143
```

28.2 ssl_ports Option Under tcp_listen

Under a top level `imapproxy`, `popproxy`, or `submitproxy` component, inside a named `tcp_listen` group, the `ssl_ports` option specifies the TCP port(s) on which that proxy server listens for SSL connections; *e.g.*:

```
msconfig> set imapproxy.tcp_listen:imap-proxy-1.ssl_ports 993
```

28.3 listen_addresses Option Under tcp_listen

Under a top level `imapproxy`, `popproxy`, or `submitproxy` component, inside a named `tcp_listen` group, the `listen_addresses` option specifies the list of interface addresses on which that proxy server listens; *e.g.*:

```
msconfig> set imapproxy.tcp_listen:imap-proxy-1.listen_addresses "192.168.1.0 190.168.1.1"
```

The list of interface address values may include hostnames, IPv4 address literals, or the string "INADDR_ANY" (which is the default).

28.4 backlog Option Under tcp_listen

RESTRICTED: Under a top level `imapproxy`, `popproxy`, or `submitproxy` component, inside a named `tcp_listen` group, the `backlog` option is intended to control the depth of the TCP backlog queue for the socket for that proxy server. However, currently this option's value is not used when set in a `tcp_listen` block, and instead the hard-coded value 1024 is used.



Part V Convergence webmail

Messaging Server includes a specialized HTTP server, [MSHTTPD](#), that provides webmail client access to [the Message Store](#).

The Oracle Communications Convergence webmail client supports S/MIME (Secure/Multipurpose Internet Mail Extension). This support has a number of [configuration options](#).

New in MS 8.0.1, MSHTTPD may call out to ICAP to perform HTML sanitization; this support has a number of [icapservice configuration options](#).



Chapter 29 MSHTTP options

29.1 enable Option Under http	29-3
29.2 enablesslport Option Under http	29-3
29.3 replayformat Option Under http	29-4
29.4 port Option Under http	29-4
29.5 allowldapaddresssearch Option	29-4
29.6 charsetvalidation Option	29-4
29.7 cookiename Option	29-4
29.8 detectcharset Option	29-4
29.9 forcenbsptospace Option	29-4
29.10 filterhiddenmailinglists Option	29-4
29.11 generatereceivedheader Option	29-5
29.12 ims5compat Option	29-5
29.13 ldapaddresssearchattrs Option	29-5
29.14 allowcollect Option	29-5
29.15 maxcollectmsglen Option	29-5
29.16 popbindaddr Option	29-5
29.17 maxldaplimit Option	29-5
29.18 nofilecache Option	29-5
29.19 rfc2231compliant Option	29-5
29.20 showunreadcounts Option	29-6
29.21 smtpauthpassword Option	29-6
29.22 smtpauthuser Option	29-6
29.23 usesentdate Option	29-6
29.24 xmailer Option	29-6
29.25 gzipattach Option	29-6
29.26 gzippedynamic Option	29-6
29.27 gzipstatic Option	29-6
29.28 plaintextconvspace Option	29-7
29.29 plaintexttabsize Option	29-7
29.30 httpproxyadmin Option	29-7
29.31 httpproxyadminpass Option	29-7
29.32 proxyport Option	29-7
29.33 cert_enable Option	29-7
29.34 cert_port Option	29-7
29.35 da_host Option	29-7
29.36 da_port Option	29-8
29.37 extrauserldapattrs Option	29-8
29.38 fullfromheader Option	29-8
29.39 ipsecurity Option	29-8
29.40 sso_enable Option	29-8
29.41 cookiedomain Option	29-8
29.42 singlesignoff Option	29-8
29.43 sso_id Option	29-9
29.44 sso_prefix Option	29-9
29.45 uwcenabled Option	29-9
29.46 uwcport Option	29-9
29.47 uwcsslport Option	29-9
29.48 uwccontexturi Option	29-9
29.49 uwchome Option	29-9
29.50 uwclogouturl Option	29-10

29.51	maxmessagesize Option Under http	29-10
29.52	maxpostsize Option	29-10
29.53	resourcetimeout Option	29-10
29.54	sessiontimeout Option	29-10
29.55	smtphost Option	29-10
29.56	smtpport Option	29-11
29.57	smtptls Option	29-11
29.58	sourceurl Option	29-11
29.59	spooldir Option	29-11
29.60	sslsourceurl Option	29-11
29.61	allowanonymouslogin Option Under http	29-11
29.62	altservice Option	29-11
29.63	domainallowed Option Under http	29-12
29.64	domainnotallowed Option Under http	29-12
29.65	enableuserlist Option Under http	29-12
29.66	forcetelemetry Option Under http	29-12
29.67	idletimeout Option Under http	29-12
29.68	maxsessions Option Under http	29-12
29.69	maxthreads Option Under http	29-12
29.70	numprocesses Option Under http	29-12
29.71	plaintextmncipher Option Under http	29-12
29.72	sslcache size Option Under http	29-13
29.73	ssl nicknames Option Under http	29-13
29.74	sslport Option Under http	29-13
29.75	sslusessl Option Under http	29-13
29.76	htmlprocessor Option	29-13
29.77	logunauthsession Option Under http	29-13
29.78	MSHTTP feedback options	29-13
29.78.1	spam Option	29-14
29.78.2	notspam Option	29-14
29.79	MSHTTP httpcharset and mailcharset options	29-14
29.79.1	af Option	29-14
29.79.2	ar Option	29-14
29.79.3	be Option	29-14
29.79.4	bg Option	29-15
29.79.5	ca Option	29-15
29.79.6	cs Option	29-15
29.79.7	da Option	29-15
29.79.8	de Option	29-15
29.79.9	el Option	29-16
29.79.10	en Option	29-16
29.79.11	es Option	29-16
29.79.12	et Option	29-16
29.79.13	eu Option	29-16
29.79.14	fi Option	29-17
29.79.15	fr Option	29-17
29.79.16	ga Option	29-17
29.79.17	gl Option	29-17
29.79.18	he Option	29-17
29.79.19	hr Option	29-18
29.79.20	hu Option	29-18
29.79.21	is Option	29-18
29.79.22	it Option	29-18

29.79.23	ja Option	29-18
29.79.24	ko Option	29-19
29.79.25	lt Option	29-19
29.79.26	lv Option	29-19
29.79.27	mk Option	29-19
29.79.28	nl Option	29-19
29.79.29	no Option	29-20
29.79.30	pl Option	29-20
29.79.31	pt Option	29-20
29.79.32	ro Option	29-20
29.79.33	ru Option	29-20
29.79.34	sk Option	29-21
29.79.35	sl Option	29-21
29.79.36	sq Option	29-21
29.79.37	sr Option	29-21
29.79.38	sv Option	29-21
29.79.39	th Option	29-22
29.79.40	tr Option	29-22
29.79.41	uk Option	29-22
29.79.42	yi Option	29-22
29.79.43	zh-cn Option	29-22
29.79.44	zh-tw Option	29-23
29.80	MSHTTP sieve options	29-23
29.80.1	port Option Under sieve	29-23
29.80.2	sslport Option Under sieve	29-23

The [http.enable](#) and/or [http.enablesslport](#) options are the fundamental options for enabling operation of the MSHTTP server. Many other options are available to further modify and tune its operation.

Under `http`, there are additional groupings of options under [feedback](#), [httpcharset](#) and [mailcharset](#), and [sieve](#).

29.1 enable Option Under http

The `enable` MSHTTP option (`service.http.enable` in legacy configuration) enables the HTTP service on `start-msg` startup. Note: HTTP over SSL service is enabled/disabled separately using [http.enablesslport](#) in Unified Configuration, or `service.http.enablesslport` in legacy configuration.

29.2 enablesslport Option Under http

Sets whether or not the HTTP over SSL service for Convergence is started. If both [http.enable](#) and `http.enablesslport` (Unified Configuration) or `service.http.enable` and `service.http.enablesslport` (legacy configuration) are turned off, then `msprobe` does not try to monitor http. If enabled, the HTTP+SSL service listens on the port specified by the [sslport](#) MSHTTP option. For the 7.0.5 release, the [sslusessl](#) option must also be explicitly set to enable the separate SSL port. For the 8.0 release, setting this option enables the separate SSL port and it is no longer necessary to explicitly set the [sslusessl](#) option.

29.3 replayformat Option Under http

The `replayformat` MSHTTPD option, `http.replayformat`, specifies the format for authentication replay from `mshttpd` to IMAP backend. Supports `%o` for original userid, `%s` for `user@domain`, `%U` for userid only, `%V` for virtual domain, `%A[attr]` for value of specified user's attribute.

29.4 port Option Under http

The `port` MSHTTPD option specifies the Messenger Express HTTP port.

29.5 allowldapaddresssearch Option

The `allowldapaddresssearch` MSHTTPD option controls whether legacy webmail client users (*i.e.*, users of Messenger Express & Communications Express) can search the directory for addresses.

29.6 charsetvalidation Option

Set the `charsetvalidation` MSHTTPD option to "0" to disable charset validation on data sent to webmail client (not recommended). Setting this to "0" is a workaround to view messages in webmail that are not labelled with the correct charset (the charset would be set then in the browser), but this will also likely generate Javascript errors and so cannot be recommended.

29.7 cookiename Option

The `cookiename` MSHTTPD option specifies the cookie name to use to pass the HTTP session ID rather than including it as part of the URL. `cookiename` defaults to `webmailsid` if the `uwcenabled` MSHTTPD option (`local.webmail.sso.uwcenabled` in legacy configuration) is enabled, and is unset if `uwcenabled` (`local.webmail.sso.uwcenabled` in legacy configuration) is not enabled.

29.8 detectcharset Option

The `detectcharset` option for `mshttpd` enables automatic character set detection for unlabeled text parts supplied by Convergence.

29.9 forcenbsptospace Option

Enabling `forcenbsptospace` will cause `mshttpd` to scan the incoming text and html parts of submitted messages for UTF-8 non-breaking spaces (0xC2A0) and replace them with ASCII spaces (0x20).

Note that this option will only function when the incoming content is UTF-8 encoded. Also note that this replacement is done blindly and may have a negative impact on space-sensitive content.

29.10 filterhiddenmailinglists Option

The `filterhiddenmailinglists` MSHTTP option excludes the `mgmanhidden` LDAP attribute from the search filter when set to 0.

29.11 generatereceivedheader Option

If the `generatereceivedheader` MSHTTP option is set to "0", webmail will not generate a Received: header, which normally contains the IP address of the sender.

29.12 ims5compat Option

Set the `ims5compat` MSHTTP option to 1 on the MEMs and the backend servers to use 5.2 Messaging Express with a 6.x MEM.

29.13 ldapaddresssearchattrs Option

The `ldapaddresssearchattrs` MSHTTP option specifies a string containing a comma-delimited list of LDAP attributes returned to legacy webmail client users (*i.e.*, users of Messenger Express or Communications Express)\ in a directory search.

29.14 allowcollect Option

Set the `allowcollect` MSHTTP option to 0 to prevent the server from performing remote POP mailbox collection.

29.15 maxcollectmsglen Option

The `maxcollectmsglen` MSHTTP option specifies the maximum message size the server collects from a remote POP mailbox. If any message in the mailbox to be collect exceeds this size, the collection will halt when that message is encountered.

29.16 popbindaddr Option

The `popbindaddr` MSHTTP option specifies the IP address to which to bind outgoing POP connections when collecting external mail. If unset, defaults to the value of [listenaddr](#).

29.17 maxldaplimit Option

The `maxldaplimit` MSHTTP option sets the maximum LDAP lookup limit.

29.18 nofilecache Option

The `nofilecache` MSHTTP option disables html files caching; used for debugging.

29.19 rfc2231compliant Option

The `rfc2231compliant` MSHTTP option enables webmail's [RFC 2231](#) encoder so that the attachment filename will be encoded in the method defined by [RFC 2231](#).

For the MTA's handling of [RFC 2231](#) encoded material, see the [parameterformat*](#) channel options.

29.20 showunreadcounts Option

The `showunreadcounts` MSHTTP option controls showing the unread message count in parentheses after the folder name. This option is only applicable for the Messenger Express and Communications Express web clients. For the Convergence web client this setting is always enabled.

29.21 smtpauthpassword Option

The `smtpauthpassword` MSHTTP option specifies the password that will be used when `mshttpd` submits mail to the MTA. See also [smtpauthuser](#).

Although this option has no default, initial configuration usually sets this option to have the value of the admin user's password.

29.22 smtpauthuser Option

When `mshttpd` submits mail to the MTA, SMTP authentication will be used if both `smtpauthuser` and [smtpauthpassword](#) are set. These two MSHTTP options specify the administrative user name and password that are used to submit mail on behalf of end users.

29.23 usesentdate Option

If the `usesentdate` MSHTTP option is set to "1", webmail will use a message's Date: header for the date the message was received. If set to "0", webmail will use the date the message arrived in the user's mailbox, which is considered more accurate.

29.24 xmailer Option

The `xmailer` MSHTTP option may be set to override the X-Mailer: header field value with the specified string.

29.25 gzipattach Option

The `gzipattach` MSHTTP option enables (when set to 1) or disables attachment download gzip by default for Internet Explorer clients.

29.26 gzipdynamic Option

The `gzipdynamic` MSHTTP option enables or disables compression of dynamic content (for example: request to *.msc files) delivered to Messenger Express or Communications Express mail clients. This can be disabled if Messenger Express or Communications Express users are getting corrupted content and cannot open their mail pages.

29.27 gzipstatic Option

The `gzipstatic` MSHTTP option enables or disables compression of static content (for example: HTML files) delivered to Messenger Express or Communications Express mail clients. This can be disabled if Messenger Express or Communications Express users are getting corrupted content and cannot open their mail pages.

29.28 plaintextconvspace Option

If the `plaintextconvspace` MSHTTP option is set to "1", spaces in text messages will be converted to non-breaking spaces in webmail.

29.29 plaintexttabsize Option

The `plaintexttabsize` MSHTTP option sets the tabsize for text message display in webmail.

29.30 httpproxyadmin Option

The `httpproxyadmin` MSHTTP option specifies a back-end store admin login name. DEPRECATED: Consider using `proxyadmin` instead; `httpproxyadmin` will be ignored if `proxyadmin` has a value.

29.31 httpproxyadminpass Option

The `httpproxyadminpass` MSHTTP option specifies a back-end store admin password. DEPRECATED: Consider using `proxyadminpass` instead; `httpproxyadminpass` will be ignored if `proxyadminpass` has been set.

29.32 proxyport Option

The `proxyport` MSHTTP option configures the port number of the back-end Messenger Express (HTTP) server with the Messaging Multiplexor.

29.33 cert_enable Option

The `cert_enable` MSHTTP option controls whether to verify certificates against a CRL. When this is set, ensure that the `crlenable` S/MIME option (in legacy configuration, the `crlenable` parameter in the `smime.conf` file) is set to 1.

29.34 cert_port Option

The `cert_port` MSHTTP option 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.

29.35 da_host Option

The `da_host` MSHTTP option specifies the iPlanet Delegated Administrator 1.x hostname. Initially set to value of the `hostname` Base option (in legacy configuration,

local.hostname). This option is available for backwards compatibility only, and should no longer be used.

29.36 da_port Option

The da_port MSHTTP option specifies the iPlanet Delegated Administrator 1.x port, by default 8080. This option is available for backwards compatibility only, and should no longer be used.

29.37 extrauserldapattrs Option

The extrauserldapattrs MSHTTP option, http.extrauserldapattrs, specifies the names of extra LDAP attributes returned to client (for customization). Syntax: attrname[:w] [,attrname]... (:w if read-write attribute).

29.38 fullfromheader Option

If the fullfromheader MSHTTP option is set, then use the cn attribute (as well as the mail attribute) from the user's LDAP entry to build the "From:" header for outgoing messages; that is, include the cn value, if there is one.

29.39 ipsecurity Option

The ipsecurity MSHTTP option sets whether or not to restrict session access to login IP addresses. If set to 1, then when the user logs in, the server remembers which IP address the user used to log in. Then it only allows that IP address to use the session cookie it issues to the user.

29.40 sso_enable Option

The sso_enable MSHTTP option enables (legacy) trusted circle single sign on functions, including accepting and verifying SSO cookies presented by the client when the login page is fetched. It returns an SSO cookie to the client for a successful login and responds to requests from other SSO partners to verify its own cookies. Setting this option to 0 (the default) disables trusted circle SSO.

29.41 cookiedomain Option

The cookiedomain MSHTTP option is a trusted circle SSO (legacy) parameter. The string value of this option is used to set the cookie domain value of all SSO cookies set by the Messenger Express HTTP server. This domain must match the DNS domain used by the Messenger Express browser to access the server. It is not the hosted domain name. This value must start with a period.

29.42 singlesignoff Option

The singlesignoff MSHTTP option is used by trusted circle SSO (legacy). When this option is set, the server will remove all single sign-on cookies for the user matching the value of

`sso_prefix` (in legacy configuration, `local.webmail.sso.prefix`). If set to 0 in this context, the server removes only its single sign-on user cookie.

29.43 sso_id Option

The `sso_id` MSHTTP option is a trusted circle SSO (legacy) parameter. The string value of this option is used as the application ID value when formatting SSO cookies set by the Messenger Express HTTP server. The default value is null. This is an arbitrary string. Its value must match what you specify for the Delegated Administrator in its `resource.properties` file. The corresponding entry in `resource.properties` would be: `Verificationurl-XXX-YYY = http://webmailhost:webmailport/VerifySSO?` Where `XXX` is the `ssl_prefix` (in legacy configuration, `local.webmail.sso.prefix`) value set above, and `YYY` is the value of `sso_id` (in legacy configuration, `local.webmail.sso.id`) set here.

29.44 sso_prefix Option

The `sso_prefix` MSHTTP option is a trusted circle SSO (legacy) parameter. It specifies the prefix value when formatting SSO cookies set by the webmail server. Only SSO cookies with this prefix value are recognized by the server; all other SSO cookies are ignored.

29.45 uwcenabled Option

The `wcenabled` MSHTTP option enables (when set to 1) or disables (when set to 0) Communications Express access to Messenger Express. When enabled, the session ID will be passed in a cookie with the name of the value of the `cookieName` MSHTTP option (Unified Configuration) or `local.service.http.cookieName` configutil parameter (legacy configuration) if such an option is set, or with the name `webmailsid` if `cookieName` (`local.service.http.cookieName` in legacy configuration) is not set. Both `wcenabled` and `cookieName` (`local.webmail.sso.wcenabled` and `local.service.http.cookieName` in legacy configuration) must match on the front and back ends.

29.46 uwcport Option

The `wcport` MSHTTP option specifies the Communications Express port.

29.47 uwcsslport Option

The `wcsslport` MSHTTP option specifies the Communications Express SSL port.

29.48 uwccontexturi Option

The `wccontexturi` MSHTTP option specifies the path in which Communications Express is deployed. Specify this parameter only when Communications Express is not deployed under `/`. For example, if Communications Express is deployed in `/uwc`, `local.webmail.sso.wccontexturi=uwc`.

29.49 uwchome Option

The uwchome MSHTTP option specifies the URL required to access the home link.

29.50 uwlogouturl Option

The uwlogouturl MSHTTP option specifies the URL Messenger Express uses to invalidate the Communications Express session.

29.51 maxmessagesize Option Under http

The maxmessagesize MSHTTP option specifies the maximum message size (in bytes) client is allowed to send through MSHTTP. Note that the [SMTP server](#) to which the message is submitted may also impose its own, separate size limit.

29.52 maxpostsize Option

The maxpostsize MSHTTP option specifies the maximum HTTP post content length, in bytes. If not specified, the default is $\max(5*1024*1024, \text{http.maxmessagesize})$. In legacy configuration this would be $\max(5*1024*1024, \text{service.http.maxmessagesize})$.

29.53 resourcetimeout Option

The resourcetimeout MSHTTP option specifies the time, in seconds, after which mshttpd flushes cached session data from memory. Lower values will use less memory and higher values incur less overhead from resynchronizing from the session database. For correct session expiration this timeout is never higher than half the [session timeout](#) (and mshttpd enforces this). The default is $\min(900, \text{sessiontimeout}/2)$, thus normally 900, (corresponding to 15 minutes), unless [sessiontimeout](#) has been set to an unusually low value.

29.54 sessiontimeout Option

The sessiontimeout MSHTTP option specifies the Webmail client session timeout in seconds. The default is 7200 (corresponding to 2 hours); setting sessiontimeout to 0 will result in a timeout value of $30*24*3600$ (corresponding to 30 days); attempting to set sessiontimeout to a positive value less than 10 will result in a value of 10 (10 seconds) being used.

29.55 smtphost Option

The smtphost MSHTTP option specifies a space-separated list of SMTP server hostnames. If smtphost is not specified, it will default to the value of the [listenaddr](#) base option (in legacy configuration, `service.listenaddr`) and if that's not set it will default to the value of the [hostname](#) base option (in legacy configuration, `local.hostname`). If none of the servers respond, a last resort attempt will be made to connect to the user's mailHost (as determined from the mailHost LDAP attribute in the user's LDAP entry).

Normally the SMTP server or SMTP SUBMIT server to which MSHTTP submits would be a Messaging Server SMTP server or SMTP SUBMIT server; see [TCP/IP channels](#) for a starting point discussion on configuration of Messaging Server SMTP and SMTP SUBMIT servers.

29.56 smtpport Option

The `smtpport` MSHTTP option specifies the SMTP or SMTP SUBMIT port to which MSHTTP will connect when submitting messages. The host(s) to which MSHTTP connects are set via the `smtphost` MSHTTP option.

Normally the SMTP server or SMTP SUBMIT server to which MSHTTP submits would be a Messaging Server SMTP server or SMTP SUBMIT server -- see [TCP/IP channels](#) for a starting point discussion on configuration of Messaging Server SMTP and SMTP SUBMIT servers. So in particular, for the port(s) on which such a Messaging Server SMTP server or SMTP SUBMIT server listens, see the `tcp_ports` Dispatcher service option, e.g., `dispatcher.service:SMTP.tcp_ports` or `dispatcher.service:SMTP_SUBMIT.tcp_ports`.

The default for `smtpport` is 25. To have MSHTTP submit to an SMTP SUBMIT server instead, note that `smtpauthuser` and `smtpauthpassword` must be set properly, and then set `http.smtpport` to 587.

29.57 smtptls Option

The `smtptls` MSHTTP option specifies whether to use TLS for SMTP connections; that is, whether MSHTTP uses the SMTP extension STARTTLS and negotiates TLS use.

29.58 sourceurl Option

The `sourceurl` MSHTTP option specifies the URL of the webmail server.

29.59 spooldir Option

The `spooldir` MSHTTP option specifies the attachment spool directory for client outgoing mail; if unspecified, uses the directory specified in the `tmpdir` option (in legacy configuration, `local.tmpdir`).

29.60 sslsourceurl Option

The `sslsourceurl` MSHTTP option specifies the URL of the webmail server when SSL is enabled.

29.61 allowanonymouslogin Option Under http

The `allowanonymouslogin` MSHTTP option enables the SASL ANONYMOUS mechanism for `mshttpd`.

29.62 altservice Option

The `mshttpd` daemon (webmail proxy) normally checks the `mailAllowedServiceAccess` and related LDAP attributes to see if the `'http' service` is enabled for that user. If the

altservice MSHTTP option is set to 1, it will instead use 'mshttpd' as the service name for such checks. This is useful if different [access control settings](#) are needed for the mshttpd daemon than for a front-end http server such as the one used by Convergence.

29.63 domainallowed Option Under http

The domainallowed MSHTTP option specifies [access filters](#) specifying which domains and/or IP addresses are allowed HTTP access.

29.64 domainnotallowed Option Under http

The domainnotallowed MSHTTP option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed HTTP access.

29.65 enableuserlist Option Under http

The enableuserlist [MSHTTP option](#) enables `imsconnutil` connected user listing for HTTP service.

29.66 forcetelemetry Option Under http

Setting the forcetelemetry MSHTTP option to 1 forces telemetry for all users. Warning: this generates a lot of data and should not be used on a production system.

29.67 idletimeout Option Under http

The idletimeout MSHTTP option specifies a timeout, in minutes, for the low-level HTTP connection (which is different from the webmail session). Lower values will use fewer socket handles and higher values cause less overhead when the client needs to recreate the connection.

29.68 maxsessions Option Under http

The maxsessions MSHTTP option specifies the maximum number of sessions per MSHTTP server process.

29.69 maxthreads Option Under http

The maxthreads MSHTTP option specifies the maximum number of threads per MSHTTP server process.

29.70 numprocesses Option Under http

The numprocesses MSHTTP option specifies the number of HTTP server processes. Note that the Watcher must be enabled for stop-msg to correctly shut down all processes if this is set to a value larger than one.

29.71 plaintextmincipher Option Under http

If the `http.plaintextmincipher` option 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.

29.72 ssl cachesize Option Under http

The `ssl cachesize` MSHTTP option specifies the number of SSL sessions to be cached by the MSHTTP server.

29.73 ssl nicknames Option Under http

The `ssl nicknames` MSHTTP option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for HTTP to offer clients if SSL/TLS enabled. Overrides for HTTP the base level `ssl nicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

29.74 ssl port Option Under http

The `ssl port` MSHTTP option, `http.ssl port`, specifies the port number for the HTTP over SSL service. The default is 8891. Note that to enable the HTTP+SSL service, the `enablessl port` and `ssl usessl` MSHTTP options must be set.

29.75 ssl usessl Option Under http

If a server certificate is installed and the `ssl usessl` MSHTTP option is not set to 0, then STARTTLS is enabled on the MSHTTP server (listening at its regular `port`).

As regards listening at a separate `ssl port`, note that for the 7.0.5 release, the `ssl usessl` option must be *explicitly* set to 1 (even though the default was 1) as well as setting `enablessl port` to enable SSL connections on a separate `ssl port`. For the 8.0 release, it is no longer necessary to explicitly set this option in order to enable SSL connections on a separate port.

29.76 html processor Option

The `html processor` MSHTTP option controls how html mail is sanitized prior to display by the Convergence webmail client. The legacy processor (value 0) is deprecated. The standard processor is used when the value is set to 1. In MS 8.0.1, setting this option to 2 enables use of the ICAP service for HTML sanitization. With the addition of the ability to use the ICAP service, the deprecated legacy processor has been completely removed and setting the option to 0 has the same effect as setting it to 1.

29.77 logunauthsession Option Under http

The `logunauthsession` MSHTTP option enables log messages from unauthenticated client sessions. Prior to turning this on, consider verifying that your logging filesystem can handle the amount of I/O possible from unauthenticated clients connecting frequently.

29.78 MSHTTP feedback options

The `http.feedback.spam` and `http.feedback.notspam` specify the e-mail addresses to which reports of spam/not-spam are to be directed.

29.78.1 spam Option

The spam [MSHTTP feedback option](#), `http.feedback.spam` (Unified Configuration) or `service.feedback.spam` (legacy configuration), specifies an email address to which to send reports of spam mail.

29.78.2 notspam Option

The notspam [MSHTTP feedback option](#), `http.feedback.notspam` (Unified Configuration) or `service.feedback.notspam` (legacy configuration), specifies an email address where to send reports of not-spam mail.

29.79 MSHTTP httpcharset and mailcharset options

Under `http` are `httpcharset` and `mailcharset` options to set the (default) character set for various languages.

29.79.1 aF Option

The `aF` option (available under `http.httpcharset` and `http.mailcharset`) sets the character set used for Afrikaans.

29.79.1.1 Use with httpcharset

Default `http` character set for Afrikaans.

29.79.1.2 Use with mailcharset

Default mail character set for Afrikaans.

29.79.2 ar Option

29.79.2.1 Use with httpcharset

Default `http` character set for Arabic.

29.79.2.2 Use with mailcharset

Default mail character set for Arabic.

29.79.3 be Option

29.79.3.1 Use with httpcharset

Default `http` character set for Byelorussian.

29.79.3.2 Use with mailcharset

Default mail character set for Byelorussian.

29.79.4 bg Option

29.79.4.1 Use with httpcharset

Default http character set for Bulgarian.

29.79.4.2 Use with mailcharset

Default mail character set for Bulgarian.

29.79.5 ca Option

29.79.5.1 Use with httpcharset

Default http character set for Catalan.

29.79.5.2 Use with mailcharset

Default mail character set for Catalan.

29.79.6 cs Option

29.79.6.1 Use with httpcharset

Default http character set for Czech.

29.79.6.2 Use with mailcharset

Default mail character set for Czech.

29.79.7 da Option

29.79.7.1 Use with httpcharset

Default http character set for Danish.

29.79.7.2 Use with mailcharset

Default mail character set for Danish.

29.79.8 de Option

29.79.8.1 Use with httpcharset

Default http character set for German.

29.79.8.2 Use with mailcharset

Default mail character set for German.

29.79.9 e1 Option

29.79.9.1 Use with httpcharset

Default http character set for Greek.

29.79.9.2 Use with mailcharset

Default mail character set for Greek.

29.79.10 en Option

29.79.10.1 Use with httpcharset

Default http character set for English.

29.79.10.2 Use with mailcharset

Default mail character set for English.

29.79.11 es Option

29.79.11.1 Use with httpcharset

Default http character set for Spanish.

29.79.11.2 Use with mailcharset

Default mail character set for Spanish.

29.79.12 et Option

29.79.12.1 Use with httpcharset

Default http character set for Estonian.

29.79.12.2 Use with mailcharset

Default mail character set for Estonian.

29.79.13 eu Option

29.79.13.1 Use with httpcharset

Default http character set for Basque.

29.79.13.2 Use with mailcharset

Default mail character set for Basque.

29.79.14 fi Option

29.79.14.1 Use with httpcharset

Default http character set for Finnish.

29.79.14.2 Use with mailcharset

Default mail character set for Finnish.

29.79.15 fr Option

29.79.15.1 Use with httpcharset

Default http character set for French.

29.79.15.2 Use with mailcharset

Default mail character set for French.

29.79.16 ga Option

29.79.16.1 Use with httpcharset

Default http character set for Irish.

29.79.16.2 Use with mailcharset

Default mail character set for Irish.

29.79.17 g1 Option

29.79.17.1 Use with httpcharset

Default http character set for Galician.

29.79.17.2 Use with mailcharset

Default mail character set for Galician.

29.79.18 he Option

29.79.18.1 Use with httpcharset

Default http character set for Hebrew.

29.79.18.2 Use with mailcharset

Default mail character set for Hebrew.

29.79.19 hr Option

29.79.19.1 Use with httpcharset

Default http character set for Croatian.

29.79.19.2 Use with mailcharset

Default mail character set for Croatian.

29.79.20 hu Option

29.79.20.1 Use with httpcharset

Default http character set for Hungarian.

29.79.20.2 Use with mailcharset

Default mail character set for Hungarian.

29.79.21 is Option

29.79.21.1 Use with httpcharset

Default http character set for Icelandic.

29.79.21.2 Use with mailcharset

Default mail character set for Icelandic.

29.79.22 it Option

29.79.22.1 Use with httpcharset

Default http character set for Italian.

29.79.22.2 Use with mailcharset

Default mail character set for Italian.

29.79.23 ja Option

29.79.23.1 Use with httpcharset

Default http character set for Japanese.

29.79.23.2 Use with mailcharset

Default mail character set for Japanese.

29.79.24 ko Option

29.79.24.1 Use with httpcharset

Default http character set for Korean.

29.79.24.2 Use with mailcharset

Default mail character set for Korean.

29.79.25 lt Option

29.79.25.1 Use with httpcharset

Default http character set for Lithuanian.

29.79.25.2 Use with mailcharset

Default mail character set for Lithuanian.

29.79.26 lv Option

29.79.26.1 Use with httpcharset

Default http character set for Latvian.

29.79.26.2 Use with mailcharset

Default mail character set for Latvian.

29.79.27 mk Option

29.79.27.1 Use with httpcharset

Default http character set for Macedonian.

29.79.27.2 Use with mailcharset

Default mail character set for Macedonian.

29.79.28 nl Option

29.79.28.1 Use with httpcharset

Default http character set for Dutch.

29.79.28.2 Use with mailcharset

Default mail character set for Dutch.

29.79.29 no Option

29.79.29.1 Use with httpcharset

Default http character set for Norwegian.

29.79.29.2 Use with mailcharset

Default mail character set for Norwegian.

29.79.30 p1 Option

29.79.30.1 Use with httpcharset

Default http character set for Polish.

29.79.30.2 Use with mailcharset

Default mail character set for Polish.

29.79.31 pt Option

29.79.31.1 Use with httpcharset

Default http character set for Portuguese.

29.79.31.2 Use with mailcharset

Default mail character set for Portuguese.

29.79.32 ro Option

29.79.32.1 Use with httpcharset

Default http character set for Romanian.

29.79.32.2 Use with mailcharset

Default mail character set for Romanian.

29.79.33 ru Option

29.79.33.1 Use with httpcharset

Default http character set for Russian.

29.79.33.2 Use with mailcharset

Default mail character set for Russian.

29.79.34 sk Option

29.79.34.1 Use with httpcharset

Default http character set for Slovak.

29.79.34.2 Use with mailcharset

Default mail character set for Slovak.

29.79.35 s1 Option

29.79.35.1 Use with httpcharset

Default http character set for Slovenian.

29.79.35.2 Use with mailcharset

Default mail character set for Slovenian.

29.79.36 sq Option

29.79.36.1 Use with httpcharset

Default http character set for Albanian.

29.79.36.2 Use with mailcharset

Default mail character set for Albanian.

29.79.37 sr Option

29.79.37.1 Use with httpcharset

Default http character set for Serbian.

29.79.37.2 Use with mailcharset

Default mail character set for Serbian.

29.79.38 sv Option

29.79.38.1 Use with httpcharset

Default http character set for Swedish.

29.79.38.2 Use with mailcharset

Default mail character set for Swedish.

29.79.39 th Option

29.79.39.1 Use with httpcharset

Default http character set for Thai.

29.79.39.2 Use with mailcharset

Default mail character set for Thai.

29.79.40 tr Option

29.79.40.1 Use with httpcharset

Default http character set for Turkish.

29.79.40.2 Use with mailcharset

Default mail character set for Turkish.

29.79.41 uk Option

29.79.41.1 Use with httpcharset

Default http character set for Ukrainian.

29.79.41.2 Use with mailcharset

Default mail character set for Ukrainian.

29.79.42 yi Option

29.79.42.1 Use with httpcharset

Default http character set for Yiddish.

29.79.42.2 Use with mailcharset

Default mail character set for Yiddish.

29.79.43 zh-cn Option

29.79.43.1 Use with httpcharset

Default http character set for Simplified Chinese.

29.79.43.2 Use with mailcharset

Default mail character set for Simplified Chinese.

29.79.44 zh-tw Option

29.79.44.1 Use with httpcharset

Default http character set for Traditional Chinese.

29.79.44.2 Use with mailcharset

Default mail character set for Traditional Chinese.

29.80 MSHTTP sieve options

The port of the web container where the Mail Filter has been deployed may be specified via an option under `http.sieve`, either [http.sieve.port](#) or [http.sieve.sslport](#).

29.80.1 port Option Under sieve

The `port` MSHTTP sieve option, `http.sieve.port`, specifies the port of the web container where the Mail Filter has been deployed.

29.80.2 sslport Option Under sieve

The `sslport` MSHTTP sieve option, `http.sieve.sslport` (Unified Configuration) or `local.webmail.sieve.sslport` (legacy configuration), specifies the the SSL port of the web container where the Mail Filter has been deployed.



Chapter 30 SMIME options

30.1 enable Option Under smime	30-1
30.2 usercertfilter Option	30-1
30.3 trustedurl Option	30-2
30.4 certurl Option	30-2
30.5 sslrootcacertsurl Option	30-2
30.6 logindn Option	30-2
30.7 loginpw Option	30-2
30.8 platformwin Option	30-3
30.9 platformmac Option	30-3
30.10 platformlinuxx86 Option	30-3
30.11 platformhpux Option	30-3
30.12 platformsolariissparc Option	30-3
30.13 alwaysencrypt Option	30-3
30.14 alwayssign Option	30-4
30.15 crlenable Option	30-4
30.16 crldir Option	30-4
30.17 crlurllogindn Option	30-4
30.18 crlurlloginpw Option	30-5
30.19 crlmappingurl Option	30-5
30.20 timestampdelta Option	30-5
30.21 crlaccessfail Option	30-5
30.22 checkoverssl Option	30-5
30.23 readsigncert Option	30-6
30.24 revocationunknown Option	30-6
30.25 sendencryptcert Option	30-6
30.26 sendencryptcertrevoked Option	30-6
30.27 sendsigncert Option	30-7
30.28 sendsigncertrevoked Option	30-7
30.29 crlusepastnextupdate Option	30-7
30.30 appletlogging Option	30-7

Several options affect S/MIME operation.

In addition to the options set under `smime`, a few MSHTTP options (set under `http`) are also relevant when using S/MIME, including `cert_enable` and `cert_port`.

30.1 enable Option Under smime

The `enable` S/MIME option controls whether the S/MIME features are available to Web Client Mail users who have permission to use them.

30.2 usercertfilter Option

The `usercertfilter` SMIME option specifies the URL filter string used to search for certificates by attributes of directory objects. It is used in conjunction with the `certurl` option. It is required when S/MIME features are enabled. Use any combination of these three email address attributes (but at least one). Example:

```
( |(mail=%e)(mailAlternateAddress=%e)(mailEquivalentAddress=%e))
```

30.3 trustedurl Option

The `trustedurl` S/MIME option specifies the URL used to search for trusted certificates used to verify user certificates. It is required when S/MIME features are enabled. Optional `logindn/loginpw` credentials for use when accessing this url can be appended in the form: `|logindn|loginpw` Both or neither must appear.

Note: the administrator must create the ldap entry for "SMIME Admin". Example:

```
ldap://mail.siroe.com:389/cn=SMIME Admin, ou=people, o=mail.siroe.com, o=mailUsers?cacertificate;binary?sub?(objectclass=certificationauthority)
```

30.4 certurl Option

The `certurl` SMIME option specifies the URL used to search for user certificates. The filter in `usercertfilter` is appended to this URL with the "%e" specifiers replaced by the email address of the user for whom certificates are being searched. It is required when S/MIME features are enabled. Optional `logindn/loginpw` credentials for use when accessing this url can be appended in the form: `|logindn|loginpw` Both or neither must appear. Example:

```
ldap://mail.siroe.com:389/ou=people, o=mail.siroe.com, o=mailUsers?userCertificate;binary?sub?
```

30.5 sslrootcacertsurl Option

The `sslrootcacertsurl` S/MIME option specifies the URL used to locate the Root certificates used for secure HTTP protocol. It is required when S/MIME features are enabled and SSL protocols are enabled. Optional `logindn/loginpw` credentials for use when accessing this url can be appended in the form: `|logindn|loginpw` Both or neither must appear. Note: the administrator must create the ldap entry for "SSL Root CA Certs". Example:

```
ldap://mail.siroe.com:389/cn=SSL Root CA Certs, ou=people, o=mail.siroe.com, o=mailQA?cacertificate;binary?base?(objectclass=certificationauthority)
```

30.6 logindn Option

The `logindn` S/MIME option specifies the Root UID for LDAP access to all the URLs specified for SMIME (options `certurl`, `crlmappingurl`, `sslrootcacertsurl`, `trustedurl`). It is optional; if not specified, the credentials of the Messaging Server are used. If specified, `loginpw` must also be specified. Note: any local credentials appended to a URL are used instead of this specification. Example: `cn=Directory Manager`

30.7 loginpw Option

The `loginpw` S/MIME option specifies the password for LDAP access to all the URLs specified for SMIME (options `certurl`, `crlmappingurl`, `sslrootcacertsurl`, `trustedurl`). It is optional; if not specified, the credentials of the Messaging Server are used. If specified,

`logindn` must also be specified. The value may be obfuscated with base64 by using `$==` instead of `==` as the `smime.conf` delimiter (this feature might appear in a future release). Note: any local credentials appended to a URL are used instead of this specification.

30.8 platformwin Option

The `platformwin` S/MIME option specifies one or more library names needed by smart cards or local key store on the Windows platform. The example below is the default; you need to specify this option only if your library settings differ. This option is required if any Communications Express Mail users use the Windows platform. Example:

```
CAPI:library=capibridge.dll;
```

30.9 platformmac Option

The `platformmac` S/MIME option specifies one or more library names needed by smart cards or local key store on the Macintosh platform. The example below is the default; you need to specify this option only if your library settings differ. This option is required if any Communications Express Mail users use the Macintosh platform. Example:

```
MOZILLA:library=libsoftokn3.dylib;
```

30.10 platformlinuxx86 Option

The `platformlinuxx86` S/MIME option specifies one or more library names needed by smart cards or local key store on the Linux platform. The example below is the default; you need to specify this option only if your library settings differ. This option is required if any Communications Express Mail users use the Linux platform. Example:

```
MOZILLA:library=libsoftokn3.so;
```

30.11 platformhpux Option

The `platformhpux` S/MIME option specifies one or more library names needed by smart cards or local key store on the HP-UX platform. The example below is the default; you need to specify this option only if your library settings differ. This option is required if any Communications Express Mail users use the HP-UX platform. Example:

```
MOZILLA:library=libsoftokn3.sl;
```

30.12 platformSolarisSparc Option

The `platformSolarisSparc` S/MIME option specifies one or more library names needed by smart cards or local key store on the Sparc Solaris platform. The example below is the default; you need to specify this option only if your library settings differ. This option is required if any Communications Express Mail users use the Sparc Solaris platform. Example:

```
MOZILLA:library=libsoftokn3.so;
```

30.13 alwaysencrypt Option

The `alwaysencrypt` S/MIME option controls the initial setting for whether all outgoing messages are automatically encrypted for all Communications Express Mail users with permission to use S/MIME. Choose one of these values:

Table 30.1 Possible alwaysencrypt option values

Value	Meaning
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.

30.14 alwaysign Option

The `alwaysign` S/MIME option controls the initial setting for whether all outgoing messages are automatically signed for all Communications Express Mail users with permission to use S/MIME. Choose one of these values:

Table 30.2 Possible alwaysign option values

Value	Meaning
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.

30.15 crlenable Option

The `crlenable` S/MIME option controls whether certificates are checked against CRLs. If there is a match, a certificate is considered revoked. The values of the `send*revoked` options (`sendencryptcertrevoked` and `sendsigncertrevoked`) determine whether a revoked certificate is rejected or used by Communications Express Mail. Choose one of these values:

Table 30.3 Possible crlenable option values

Value	Meaning
0	each certificate is not checked against a CRL.
1	each certificate is checked against a CRL. This is the default.

30.16 crldir Option

The `crldir` S/MIME option specifies the directory path information to locate the database where CRL information is stored. The default value is `SERVERROOT/data/store/mboxlist`.

30.17 crlurllogindn Option

The `crlurllogindn` S/MIME option specifies the Root UID for LDAP access to all the URLs specified in the CRL Mapping table. It is optional; if not specified, the credentials of the Messaging Server are used. If specified, `crlurlloginpw` must also be specified. Note:

any local credentials appended to a URL in the CRL Mapping table are used instead of this specification. Example: `cn=Directory Manager`

30.18 crlurlloginpw Option

Specifies the password for LDAP access to all the URLs specified in the CRL Mapping table. It is optional; if not specified, the credentials of the Messaging Server are used. If specified, `crlurllogindn` must also be specified. The value may be obfuscated with base64 by using `$==` instead of `==` as the `smime.conf` delimiter (this feature might appear in a future release). Note: any local credentials appended to a URL in the CRL Mapping table are used instead of this specification.

30.19 crlmappingurl Option

The `crlmappingurl` S/MIME option specifies the URL used to locate the CRL mapping. Optional `logindn/loginpw` credentials for use when accessing this url can be appended in the form `"|logindn|loginpw"`. Both or neither must appear. Note: the administrator must create the ldap entry for "SMIME Admin". Example: `ldap://mail.siroe.com:389/cn=SMIME Admin, ou=people, o=mail.siroe.com, o=mailQA?msgCRLMappingRecord?sub?(objectclass=msgCRLMappingTable)`

30.20 timestampdelta Option

The `timestampdelta` S/MIME option specifies the time interval in seconds before the received time on messages.

30.21 crlaccessfail Option

The `crlaccessfail` S/MIME option specifies the time interval to wait after multiple CRL attempts, in the form: `"#fails:#mins:#minswait"` where:

Table 30.4 crlaccessfail fields

Field	Meaning
<code>#fails</code>	number of CRL URL failures before triggering wait
<code>#mins</code>	number of minutes during which the <code>#fails</code> must occur
<code>#minswait</code>	number of minutes to wait until another CRL URL attempt will be allowed If <code>crlaccessfail</code> is used, no fields are optional, and all values must be greater than 0.

Example: `2:20:10`.

30.22 checkoverssl Option

The `checkoverssl` S/MIME option controls whether an SSL communications link is used when checking a certificate against a CRL. Choose one of these values:

Value	Meaning
-------	---------

0	do not use an SSL communications link.
1	use an SSL communications link. This is the default.

30.23 readsigncert Option

The `readsigncert` S/MIME option controls whether the certificate used to sign a message is checked against a CRL when the message is read. Choose one of these values:

Table 30.5 `readsigncert` fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

30.24 revocationunknown Option

The `revocationunknown` S/MIME option 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 if the certificate is valid or has a revoked status. Choose one of these values:

Value	Meaning
ok	treat the certificate as valid.
revoked	treat the certificate as revoked. This is the default.

30.25 sendencryptcert Option

The `sendencryptcert` S/MIME option controls whether a certificate used to encrypt an outgoing message is checked against a CRL before using it. Choose one of these values:

Table 30.6 `sendencryptcert` fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

30.26 sendencryptcertrevoked Option

The `sendencryptcertrevoked` S/MIME option determines the action to take if a certificate used to encrypt an outgoing message has been revoked. Choose one of these values:

Table 30.7 `sendencryptedcertrevoked` fields

Value	Meaning
allow	use the certificate.
disallow	do not use the certificate. This is the default.

30.27 sendsigncert Option

The `sendsigncert` S/MIME option controls whether a certificate used for signing an outgoing message is checked against a CRL before using it. Choose one of these values:

Table 30.8 `sendsigncert` fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

30.28 sendsigncertrevoked Option

The `sendsigncertrevoked` S/MIME option determines the action to take if the certificate used to sign an outgoing message has been revoked. Choose one of these values:

Table 30.9 `sendsigncertrevoked` fields

Value	Meaning
allow	use the certificate.
disallow	do not use the certificate. This is the default.

30.29 crlusepastnextupdate Option

The `crlusepastnextupdate` S/MIME option specifies whether to continue to use a CRL after its next update date (in case a new CRL is not available or accessible). Choose one of these values:

Table 30.10 `crlusepastnextupdate` fields

Value	Meaning
0	do not use a CRL after its next update date.
1	continue to use a CRL after its next update date. This is the default.

30.30 appletlogging Option

The `appletlogging` S/MIME option specifies whether to enable logging on the applet. Choose one of these values:

Table 30.11 `appletlogging` fields

Value	Meaning
0	do not log applet output. This is the default.
1	log applet output.



Chapter 31 SSO options

31.1 verifyurl Option	31-1
-----------------------------	------

The `verifyurl` option is the only option under the `sso` group. See also various [MSHTTP options](#) relating to SSO, especially the `sso_*` MSHTTP options.

31.1 `verifyurl` Option

The `verifyurl` option specifies a trusted circle SSO (legacy) parameter. It sets the verify URL values for peer SSO applications. The standard form of the value of the verify URL is:

```
http://[peer_hostname]:[port]/VerifySSO?
```

This value should be set for the application ID of a peer SSO application whose SSO cookies are to be honored.



Chapter 32 icapservice options

32.1 forcetelemetry Option Under icapservice	32-1
32.2 service_name Option	32-1
32.3 server_host Option Under icapservice	32-1
32.4 server_port Option Under icapservice	32-1

New in MS 8.0.1, if the [htmlprocessor](#) MSHTTP option is set to 2, then the ICAP service will be used to perform HTML sanitization. There are a few options that configure this use of the ICAP service.

32.1 forcetelemetry Option Under icapservice

Setting the `forcetelemetry` `icapservice` option to 1 forces telemetry for all users. Warning: this generates a lot of data and should not be used on a production system.

32.2 service_name Option

The `service_name` option sets the name used by the ICAP client when performing HTML sanitization. The default is "email" and there's no need to change this.

32.3 server_host Option Under icapservice

The `server_hosticapservice` option specifies the ICAP server host name. If empty or not set, the loopback interface is used.

32.4 server_port Option Under icapservice

The `server_porticapservice` option specifies the ICAP server port. The default is 1344.



Part VI The MTA

The Messaging Server MTA is a general-purpose, store-and-forward system for distributing computer-based mail. The term *store-and-forward* means that the Messaging Server MTA automatically handles the queuing and retransmission of mail messages necessitated when network links or services are temporarily unavailable. In contrast to *mail user agents* (MUAs) such as Messenger Express or Communications Express (UWC) which are used to create and read electronic mail messages, the Messaging Server MTA is a *mail transport agent* (MTA) responsible for directing messages to the appropriate network transport and ensuring reliable delivery over that transport.

The MTA provides a uniform distribution environment that can be interfaced to multiple user interfaces (MUAs), networks, protocols, and transport mechanisms. As this interfacing, from the user's point of view, is accomplished transparently, the MTA presents to the user a homogeneous mail network; *i.e.*, the MTA seamlessly blends heterogeneous mail networks into a single, coherent mail system.

There are many components to the MTA.



Chapter 33 Channels

33.1 Available channels	33-2
33.2 Channel configuration	33-5
33.3 Channel options	33-6
33.3.1 Alphabetic list of channel options	33-7
33.3.2 Functional group list of channel options	33-14
33.3.3 Addresses channel options	33-28
33.3.4 Attachments and MIME processing channel options	33-45
33.3.5 BSMTP-specific channel options	33-51
33.3.6 Character sets and eight bit data channel options	33-52
33.3.7 Conversion tag and service conversion channel options	33-55
33.3.8 Display label channel options	33-55
33.3.9 DKIM channel options	33-56
33.3.10 Error interpretation channel options	33-57
33.3.11 File creation in the MTA queue area channel options	33-57
33.3.12 Gateway or firewall or mailhub channel options	33-59
33.3.13 Headers channel options	33-62
33.3.14 Host name channel options	33-79
33.3.15 Incoming channel match and switch channel options	33-81
33.3.16 Logging and debugging channel options	33-84
33.3.17 Long address lists or headers channel options	33-86
33.3.18 Message hash channel options	33-90
33.3.19 Message tracking channel options	33-91
33.3.20 MLS channel options	33-93
33.3.21 Notification messages and postmaster messages channel options	33-93
33.3.22 Processing control and job submission channel options	33-99
33.3.23 Sensitivity limits channel options	33-107
33.3.24 Sieve filters and delivery flags channel options	33-108
33.3.25 Size limits on messages channel options	33-111
33.3.26 Spamfilter channel options	33-115
33.3.27 SMTP and LMTP protocol channel options	33-116
33.3.28 TCP/IP connections and DNS lookups channel options	33-135
33.3.29 TLS and SASL channel options	33-146
33.4 Header option files	33-155
33.4.1 Header option file location	33-156
33.4.2 Header option file format	33-156

The MTA consists of a large number of components, but the central unifying construct in the MTA is the channel. A channel represents an e-mail 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. Only the MTA administrator need know anything about the MTA's channels; users are never aware of the existence of channels and only see a single, uniform interface regardless of how messages reach their destination.

Each channel consists of one or more channel programs and an outgoing message queue for storing messages that are destined to be sent to one or more of the systems associated with the channel. Channel programs perform two functions: (1) they transmit messages to remote systems, deleting them from their queue after they are sent, and (2) they accept messages from remote systems, placing them in the channel queues. Note that while a channel program only removes messages from its own queue it can enqueue messages on any queue whatsoever, including its own.

A channel program which initiates a transfer to a remote system on its own is called a "master" program, while a program which accepts transfers initiated by a remote system is called a "slave" program. A channel may be served by a master program, a slave program, or both. Either type of program may or may not be [bidirectional](#); the direction in which a message is travelling may have nothing to do with the type of program that handles it. Very often, however, a master program transmits messages, while a slave program receives messages. An SMTP channel, for instance, has a master program that only transmits messages (the SMTP client) and a slave program that only receives messages (the SMTP server).

33.1 Available channels

Every MTA channel has a unique name containing up to 32 characters. Only lowercase letters, numbers, underscores, and dollar signs should be used in channel names.

Certain channel names are reserved for particular uses. Moreover, the MTA (especially the [Job Controller](#)) recognizes certain families of channel names, (channel name prefixes), and will make internal assumptions about such channels. There are both hard-coded expectations of certain special channel name usage, plus in particular some historical but now wide-spread "conventions" regarding use of certain `tcp_*` channel names. Using channel names in a conflicting manner can lead to serious problems. MTA administrators are encouraged to use these channels for the stated purposes and in general to pick channel names of their own that do not conflict with these usage conventions.

Table 33.1 Modern reserved channel names

Name	Reserved For
<code>bitbucket</code>	Bit bucket channel (deletes all messages queued to it)
<code>bsin_*</code>	BSMTP inbound channels
<code>bsout_*</code>	BSMTP outbound channels
<code>circuitcheck</code>	Message circuit checking channel
<code>conversion</code>	Message body part conversion channel
<code>defragment</code>	Message defragmentation channel
<code>filter_discard</code>	Channel for delayed deletion of message discarded or jettisoned due to Sieve actions (or analogous <code>*_ACCESS</code> mapping table flag effects)
<code>hold</code>	Channel where messages are temporarily detained for administrative purposes such as user migration
<code>ims-ms</code>	Channel delivering to the Message Store (without use of LMTP; see also the tcp_lmtpss channel when LMTP is used instead).
<code>ims-ms_*</code>	Additional channels delivering to the Message Store.
<code>l</code>	The local channel; in modern Messaging Server configurations, a "placeholder" channel that performs no delivery but rather performs alias processing only.
<code>native</code>	Delivery to UNIX <code>/var/mail</code> mailboxes.
<code>pipe</code>	Pipe channel
<code>pipe_*</code>	Additional pipe channels
<code>process</code>	Processing channel

process_*	Additional processing channels
reprocess	Reprocessing channel
reprocess_	Additional reprocessing channels
sms*	SMS channels
tcp_auth	By convention, the SMTP-over-TCP/IP channel handling incoming authenticated messages.
tcp_intranet	By convention, the SMTP-over-TCP/IP channel communicating with other internal hosts.
tcp_lmtpcs*	By convention, LMTP client channels.
tcp_lmtpss	By convention, the LMTP server delivering to the Message Store on an LMTP back end system.
tcp_local	By convention, the SMTP-over-TCP/IP channel communicating with the Internet.
tcp_submit	By convention, the SMTP SUBMIT server channel.
tcp_tas	By convention, the SMTP-over-TCP/IP channel receiving telephony messages that need so-called "Guaranteed Message Deposit" (quota bypass for message delivery).
tcp_*	Additional TCP/IP SMTP (or LMTP) channels
test_smtp_*	Test (sample code) channels
uucp_	UUCP channel (UNIX, or DEC/Shell UUCP)

In addition to the above modern channel names, there are various obsolete or historical channels where use of those old channel names may cause confusion:

Table 33.2 Outdated reserved channel names

Name or prefix	Reserved For
address	Extract addressing information from the body of a message
address_*	Additional addressing channels
aoce_	Apple AOCE channels
anje_	ANJE (BITNET)
bit_	Jnet (BITNET)
bull_	BULLETIN channels
cc_	cc:Mail channels
cn_	Internal usage by the national Australian network
ctcp_	Carnegie Mellon University TCP/IP channels; obsolete
d	The DECnet MAIL channel; used to deliver messages across DECnet via VMS MAIL
d_	Additional MAIL-11 over DECnet channels
data_to_bitmap	Raw FAX data to bitmap channel
data_to_bitmap_	Additional data to bitmap channels
directory	Directory alias expansion channel
directory_	Directory alias expansion channels

Available channels

dn_	PhoneNet over DECnet
dsmtplib_	SMTP over DECnet
era_	ERA channels
etcp_	Excelan TCP/IP channels; obsolete
faxsr_	Fax Sr. channels
fax_to_data	Inbound FAX to raw data channel
fax_to_data_	Inbound FAX to raw data channel
ff_	Microsoft® Mail channels
ftcp_	Network Research Corporation FUSION TCP/IP channels; obsolete
g3_to_fax	Group 3 to FAX modem spooler
g3_to_fax_	Group 3 to FAX modem channels
ker_	Kermit protocol
ln_	Lotus Notes channels
mail_	General VMS MAIL delivery
mailserv	Mail and list server channel
mhs_	Novell MHS channels
mime_to_x400	MIME to X.400 conversion channel
mime_to_x400_	MIME to X.400 conversion channels
mime_to_x40084_	MIME to X.400-1984 conversion channels; obsolete
mint	MINT user agent from Wesleyan University
mr_	PMDF-MR gateway
mrif_	PMDF-MR as Message Router TS replacement channels
msgstore	PMDF Message Store delivery channel
msgstore_	PMDF MessageStore channel
mtcp_	Process Software MultiNet (formerly Cisco MultiNet, formerly TGV MultiNet) TCP/IP channels; obsolete
netdata_	Netdata (PROFS) channels
notes_	VAX NOTES channels
osfl_	UNIX local channels
ovvm_	OV/VM (PROFS) channels
p	Generic PhoneNet channel; used to communicate with a central PhoneNet host
p_	PhoneNet channels
pager	E-mail to pager channel
pager_	Pager channels
popstore	PMDF popstore delivery channel; a msgstore channel can be--- and typically is -- used instead
profs_	PROFS channels

printer	e-mail to spooled printer
printer_*	Additional e-mail to spooled printer channels
ps_to_g3	PostScript to Group 3 FAX interpreter
ps_to_g3_	PostScript to Group 3 FAX channels
ptcp_	Process Software TCPware
px25_	PhoneNet over X.25; obsolete
qm_	QuickMail channels
snads_	SNADS channels
subject	Channel to extract information from Subject: lines
sync_db_	Database synchronization channels
sync_dirbot_	Directory synchronization robot (DIRBOT) channels
sync_ldap_	LDAP directory synchronization channels
sync_ldif_	LDIF directory agent channels
sync_ln_	Lotus Notes directory agent channels
text_to_ps	Text to PostScript converter
text_to_ps_	Additional text to PostScript channels
utcp_	ULTRIX (UCX) Connection TCP/IP channels; obsolete
vn_	UUCP channel (DECUS UUCP)
wpo_	GroupWise (WordPerfect Office) channels
wtcp_	Wollongong TCP/IP (WIN/TCP) channels; obsolete
x400_	X.400 channels
x40084_	X.400-1984 channels; obsolete
x400_to_mime	X.400 to MIME conversion channel
x400_local	X.400 transport channel
xapi_	MAILbus 400 channels
xsmtplib_	SMTP over X.25; obsolete

33.2 Channel configuration

In Unified Configuration, each MTA [channel](#) is configured as a named set of [options](#) under a channel group. For instance, configuration of the `tcp_local` channel (the typical channel used to communicate with the Internet) could be along the lines of:

```
msconfig> show channel:tcp_local
role.channel:tcp_local.official_host_name = tcp-daemon
role.channel:tcp_local.identnonnumeric (novalue)
role.channel:tcp_local.inner (novalue)
role.channel:tcp_local.loopcheck (novalue)
role.channel:tcp_local.maysaslserver (novalue)
role.channel:tcp_local.maytlserver (novalue)
role.channel:tcp_local.defaultmx (novalue)
role.channel:tcp_local.pool = SMTP_POOL
```

```
role.channel:tcp_local.remotehost (novalue)
role.channel:tcp_local.saslswitchchannel = tcp_auth
role.channel:tcp_local.single_sys (novalue)
role.channel:tcp_local.smtp (novalue)
role.channel:tcp_local.switchchannel (novalue)
```

In legacy configuration, each MTA channel was configured in the `imta.cnf` file, with blank lines separating distinct channel definitions. For instance, a `tcp_local` channel definition in `imta.cnf` might appear as (with a blank line before and after these lines):

```
tcp_local smtp mx single_sys identnonnumeric pool SMTP_POOL \
  switchchannel maysaslserver saslswitchchannel tcp_auth maytlserver \
  inner loopcheck remotehost
tcp-daemon
```

Note that in Unified Configuration, the `edit channels` command may be used to edit channel definitions "as if" they were in the familiar, legacy configuration file form:

```
msconfig> edit channels
```

Note that the channel-specific options -- those set in legacy configuration in channel option files, rather than as legacy configuration channel keywords -- in Unified Configuration are set under the options named group, *e.g.*,

```
msconfig> set channel:tcp_local.options.ALLOW_ETRNS_PER_SESSION 2
```

Since channel-specific options are specific to the type of channel in question -- different types of channels tend to have completely different channel-specific options -- look under the discussions of the [specific channel type](#) to find out details of what channel-specific options are available for a particular channel type, as channel-specific options do not appear under the general list of options.

33.3 Channel options

Channel options are available to adjust many attributes and aspects of channel operation. Unified Configuration channel options subsume the *channel keywords* of legacy configuration, as well as the second and optional additional lines of legacy configuration channel definitions; and the Unified Configuration options named group allows setting the channel-specific options of legacy configuration (those set in legacy configuration in a channel option file). That is, in legacy configuration, channel keywords would appear after the channel name on the first line of the channel definition in `imta.cnf`, and then the second and optional additional lines of a channel definition in `imta.cnf` would set the [official_host_name](#) for a channel and optionally the [local_host_alias](#) and [additional_host_names](#); while channel-specific options would be set in the `channel-name_option` channel option file.

In Unified Configuration, any desired channel options are set under a named channel group; *e.g.*, for options that merely need to be turned on:

```
msconfig> set channel:channel-name.option-name
```

or for channel options that take an argument:

```
msconfig> set channel:channel-name.option-name option-value
```

Some channel options take arguments; each option argument is limited in general to 40 characters, though some special options allow arguments of 256 characters (252 characters in iMS 5.2 and earlier), and a few options (as of 7.0.5 including [sourcefilter](#) and [destinationfilter](#)) allow arguments up to 1024 characters.

In Unified Configuration, option argument case is preserved even if not quoted. Quoting is not necessary for most option arguments; but options that take multiple "arguments" (in legacy configuration terms), must in Unified Configuration be set by specifying a single, quoted argument. For instance:

```
msconfig> set channel:tcp_local.saslswitchchannel tcp_auth
role.channel:tcp_local.saslswitchchannel = tcp_auth
msconfig# set channel:tcp_local.backoff "PT30M PT1H PT1H PT2H PT4H PT8H"
role.channel:tcp_local.backoff = PT30M PT1H PT1H PT2H PT4H PT8H
```

In legacy configuration, option arguments are normally forced to lowercase, but case will be preserved if the option argument is quoted. Also, while legacy configuration options are normally limited to taking at most five arguments, when the arguments themselves are quoted that restriction is lifted; (see for instance, the [backoff](#) channel option).

Note that the channel-specific options -- those set in legacy configuration in channel option files, rather than as legacy configuration channel keywords -- in Unified Configuration are set under the options named group, *e.g.*,

```
msconfig> set channel:tcp_local.options.ALLOW_ETRNS_PER_SESSION 2
```

Since channel-specific options are specific to the type of channel in question -- different types of channels tend to have completely different channel-specific options -- look under the discussions of the [specific channel types](#) to find out details of what channel-specific options are available, as channel-specific options do not appear under the general list of options, nor as channel options.

33.3.1 Alphabetic list of channel options

[Channel options listed alphabetically](#) below lists channel keywords alphabetically. Channel options shown in **bold face type** are defaults.

Table 33.3 Channel options listed alphabetically

Option	Usage
_733	Use % routing in the envelope; synonymous with percents
_822	Use source routes in the envelope; synonymous with sourceroute
acceptalladdresses	(New in JES MS 6.1) Accept messages despite certain errors that would normally cause message rejection
acceptvalidaddresses	(New in JES MS 6.1) Perform normal rejection checks on incoming messages
additional_host_names	Additional hosts the channel can reach
addlineaddr	RESTRICTED: Attempt to extract additional envelope recipient addresses from X-VMS-To: and X-VMS-Cc: header lines
addresssrs	(New in MS 6.3p1) Addresses matching this channel are eligible for SRS encoding
addreturnpath	Add a Return-Path: header line
addrspfile	Number of addresses per message file
addrspjob	Number of addresses to be processed by a single job

Alphabetic list of channel options

addrtypescan	(New in MS 7.0.5) Store recipient address "type" in an envelope flag
addrtypescanbccdefault	(New in MS 7.0.5) Store recipient address "type" in an envelope flag, assuming unmatched recipient addresses are Bcc: addresses
affinitylist	(New in MS 8.0) Enable affinity lookups, disable MX lookups
after	Specify time delay before master channel programs run
aliasdetourhost	Specify an "override" mailHost for any user found in LDAP; effect is to "detour" messages for such a user to the specified host
aliaslocal	Look up aliases; e.g., query alias file and alias database , and perform alias_url* lookups
aliasmagic	(New in JES MS 6.0) RESTRICTED: Destination channel override of the alias_magic MTA option, controlling the order of the different types of alias lookups
aliasoptindetourhost	(New in JES MS 6.2p4) Specify an "override" mailHost for any user who is opted-in via whatever LDAP attribute is named by the ldap_detourhost_optin MTA option; the effect is to "detour" messages for such a user to the specified host
aliaspostmaster	Redirect postmaster messages to the local channel postmaster
aliaswild	Do an * lookup if no exact alias match is found
allowetrn	Honor SMTP client ETRN commands
allowswitchchannel	Allow switching to this channel from a switchchannel channel
alternateblocklimit	Divert messages that exceed the specified number of blocks to the alternatechannel
alternatechannel	Messages that exceed a channel's alternateblocklimit , alternatelinelimit , or alternaterecipientlimit will be diverted to the channel's specified alternatechannel
alternatelinelimit	Divert messages that exceed the specified number of lines to the alternatechannel
alternaterecipientlimit	Divert messages that exceed the specified number of recipients to the alternatechannel
authpassword	(New in MS 7.0.5) Password for SMTP channel's client use of SMTP AUTH PLAIN
authrewrite	Use SMTP AUTH information in header
authusername	(New in MS 7.0.5) Username for SMTP channel's client use of SMTP AUTH PLAIN
autosecretary	RESTRICTED: Not yet implemented
backoff	Channel delivery retry backoff intervals
bangonly	Source channel: disable interpretation of % host-routing
bangoverpercent	Group A!B%C as A!(B%C)
bangstyle	Use UUCP ! routing in the envelope; synonymous with uucp
bidirectional	Channel is served by both a master and slave program
binaryclient	(New in MS 6.3.) RESTRICTED: Not yet fully implemented. Enable BINARYMIME support in the SMTP client
binaryserver	(Introduced in MS 6.3 but at that time RESTRICTED as not yet fully implemented; actual implementation new in MS 8.0) Enable BINARYMIME support in the SMTP server
blocketrn	Do not honor SMTP client ETRN commands
blocklimit	Maximum number of MTA blocks allowed per message
cacheeverything	Cache all connection information
cachefailures	Cache only connection failure information
cachesuccesses	Cache only connection success information
caption	(New in MS 6.3) Channel caption: a short description, suitable as the caption for a column of a table
channelfilter	Specify the location of channel filter file; synonym for destinationfilter
charset7	Default character set to associate with 7-bit text messages
charset8	Default character set to associate with 8-bit text messages
charsetesc	Default character set to associate with text containing the escape character
checkehlo	Check the SMTP greeting banner for whether to use EHLO
chunkingclient	(New in MS 6.3) Enable CHUNKING support in the SMTP client
chunkingserver	(New in MS 6.3) Enable CHUNKING support in the SMTP server
commentinc	Leave comments in message header lines intact
commentmap	Apply COMMENT_STRINGS mapping to comments in message header lines
commentomit	Remove comments from message header lines
commentstrip	Remove problematic characters from comment field in message header lines
commenttotal	Strip comments (material in parentheses) everywhere
conditionalpassthrough	"Pass-through" mode, if any Received: header lines are present
conditionalrelay	"Relay" mode, if any Received: header lines are present
conditionalsecuritymultipart	Process inside security multipart, retaining preamble material
connectalias	Do not rewrite addresses upon message dequeue
connectcanonical	Rewrite addresses upon message dequeue
contchar	DEPRECATED: Specify batch SMTP continuation line character
contposition	DEPRECATED: Specify folding point in batch SMTP lines

converttoctetstream	Convert application/octet-stream material as appropriate
copysendpost	Send copies of failures to the postmaster unless the originator address is blank
copywarnpost	Send copies of warnings to the postmaster unless the originator address is blank
daemon	Specify name of a gateway daemon (host) to route to
datefour	Convert date/time specifications to four digit years
datetwo	Convert date/time specifications to two digit years
dayofweek	Include day of week in date/time specifications
defaultthost	Specify a domain name to use to complete addresses
defaultmx	Channel determines whether or not to do MX lookups from network
defaultnameservers	Consult TCP/IP stack's choice of nameservers
deferralrejectlimit	New in MS 6.2. Limit the number of bad (failing) recipient addresses
deferred	Honor deferred delivery dates in Deferred-delivery: header lines; as of MS 7.0, deprecated in favor of deferreddestination
deferreddestination	(New in MS 7.0) Honor deferred delivery dates in Deferred-delivery: header lines; synonym for deferred
deferredsource	(New in MS 7.0) Honor deferred delivery dates in Deferred-delivery: header lines
defragment	Reassemble any MIME-compliant message/partial parts queued to this channel
deletemessagehash	(New in JES MS 6.3) Delete message hash
deliveryflags	Set flags controlling certain delivery behaviors
dequeue_removeoute	Alias for dequeue_removeoute ; in legacy configuration, strip source route (@source-route:address) when dequeuing message
dequeue_removeoute	(Unified Configuration only) Strip source route (@source-route:address) when dequeuing message
description	Channel description
destinationbrightmail	Deprecated synonym for destinationsspamfilter and (in JES MS 6.2) destinationsspamfilter1
destinationbrightmailoptin	Deprecated synonym for destinationsspamfilteroptin and (in JES MS 6.2) destinationsspamfilteroptin
destinationconversiontag	(New in MS 7.0.5.) Conversion tags to add to outgoing recipients
destinationdkimignore	(New in MS 8.0.) Take no special action in regards to DKIM-Signature: header fields
destinationdkimpreserve	(New in MS 8.0) Don't rewrite DKIM-signed messages for specified domains
destinationdkimremove	(New in MS 8.0) Remove DKIM signatures
destinationfilter	Specify the location of channel Sieve filter to apply to outgoing messages
destinationflowthrough	RESTRICTED: Not yet implemented
destinationmosolicit	New in MS 6.2. List of solicitation types not accepted
destinationsspamfilter	Synonym for destinationsspamfilter1
destinationsspamfilter1	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 1) for messages enqueued to this destination channel
destinationsspamfilter1optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 1) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter2	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 2) for messages enqueued to this destination channel
destinationsspamfilter2optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 2) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter3	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 3) for messages enqueued to this destination channel
destinationsspamfilter3optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 3) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter4	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 4) for messages enqueued to this destination channel
destinationsspamfilter4optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/filter package 4) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter5	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 5) for messages enqueued to this destination channel
destinationsspamfilter5optin	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 5) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter6	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 6) for messages enqueued to this destination channel
destinationsspamfilter6optin	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 6) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter7	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 7) for messages enqueued to this destination channel
destinationsspamfilter7optin	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 7) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilter8	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 8) for messages enqueued to this destination channel
destinationsspamfilter8optin	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 8) for messages enqueued to this destination channel, with the optin value provided as argument
destinationsspamfilteroptin	Synonym for destinationsspamfilter1optin
destinationrsrs	(New in MS 6.3p1) Messages destined out this channel are eligible for SRS encoding
disabledestinationbrightmail	Deprecated synonym for disabledestinationsspamfilter and (in JES MS 6.2) disabledestinationsspamfilter1

Alphabetic list of channel options

disabledestinationfilter	(New in MS 7.0u3) Disable evaluation and application of specified Sieve filters
disabledestinationspamfilter	Synonym for disabledestinationspamfilter1
disabledestinationspamfilter1	(New in JES MS 6.2) Disable spam/virus filtering (via spam/filter package 1) for messages enqueued to this channel
disabledestinationspamfilter2	(New in JES MS 6.2) Disable spam/virus filtering (via spam/filter package 2) for messages enqueued to this channel
disabledestinationspamfilter3	(New in JES MS 6.2) Disable spam/virus filtering (via spam/filter package 3) for messages enqueued to this channel
disabledestinationspamfilter4	(New in JES MS 6.2) Disable spam/virus filtering (via spam/filter package 4) for messages enqueued to this channel
disabledestinationspamfilter5	(New in MS 6.3) Disable spam/virus filtering (via spam/filter package 5) for messages enqueued to this channel
disabledestinationspamfilter6	(New in MS 6.3) Disable spam/virus filtering (via spam/filter package 6) for messages enqueued to this channel
disabledestinationspamfilter7	(New in MS 6.3) Disable spam/virus filtering (via spam/filter package 7) for messages enqueued to this channel
disabledestinationspamfilter8	(New in MS 6.3) Disable spam/virus filtering (via spam/filter package 8) for messages enqueued to this channel
disableetrn	Disable support for the ETRN SMTP command
disablesourcebrightmail	(New in JES MS 6.1) Deprecated synonym for disablesourcespamfilter and (in JES MS 6.2) disablesourcespamfilter1
disablesourcefilter	(New in MS 7.0u3) Disable evaluation and application of specified Sieve filters
disablesourcespamfilter	(New in JES MS 6.1) Synonym for disablesourcespamfilter1
disablesourcespamfilter1	(New in JES MS 6.2) Disable spam/virus filtering (via spam/virus filter package 1) for messages enqueued by this channel
disablesourcespamfilter2	(New in JES MS 6.2) Disable spam/virus filtering (via spam/virus filter package 2) for messages enqueued by this channel
disablesourcespamfilter3	(New in JES MS 6.2) Disable spam/virus filtering (via spam/virus filter package 3) for messages enqueued by this channel
disablesourcespamfilter4	(New in JES MS 6.2) Disable spam/virus filtering (via spam/virus filter package 4) for messages enqueued by this channel
disablesourcespamfilter5	(New in MS 6.3) Disable spam/virus filtering (via spam/virus filter package 5) for messages enqueued by this channel
disablesourcespamfilter6	(New in MS 6.3) Disable spam/virus filtering (via spam/virus filter package 6) for messages enqueued by this channel
disablesourcespamfilter7	(New in MS 6.3) Disable spam/virus filtering (via spam/virus filter package 7) for messages enqueued by this channel
disablesourcespamfilter8	(New in MS 6.3) Disable spam/virus filtering (via spam/virus filter package 8) for messages enqueued by this channel
disconnectbadauthlimit	New in MS 6.2. Force SMTP session disconnect after a specified number of failed SMTP AUTH attempts
disconnectbadburlimit	New in MS 7.0u4. Force SMTP SUBMIT session disconnect after a specified number of invalid BURL commands is exceeded
disconnectbadcommandlimit	New in MS 6.2. Force SMTP session disconnect after a specified number of bad (unrecognized) SMTP commands is exceeded
disconnectcommandlimit	New in MS 7.0u1. Force SMTP session disconnect after a specified number of SMTP commands is exceeded
disconnectrecipientlimit	New in MS 6.2. Force SMTP session disconnect after a specified number of recipients is exceeded
disconnectrejectlimit	New in MS 6.2. Force SMTP session disconnect after a specified number of bad recipients (rejected recipients) is exceeded
disconnecttransactionlimit	(New in JES MS 6.2) Force SMTP session disconnect after a specified transaction limit is exceeded
dispositionchannel	New in MS 6.2. Channel to use when generating message disposition notifications
dkimignore	(New in MS 7.0.5) Take no special action in regards to DKIM-Signature: header fields
dkimpreserve	(New in MS 7.0.5) Don't rewrite DKIM-signed messages for specified domains
dkimremove	(New in MS 7.0.5) Remove DKIM signatures
domainetrn	Honor only those SMTP client ETRN commands that specify a domain
domainvrfy	Issue SMTP VRFY commands using full address
dropblank	Strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
ehlo	Use EHLO on all initial SMTP connections
eightbit	Channel supports eight bit characters
eightnegotiate	Channel should negotiate use of eight bit transmission if possible
eightstrict	Channel should reject messages that contain unnegotiated eight bit data
enqueue_removeoute	Strip source route (@source-route:address) when enqueueing message
errsendpost	Send copies of failures to the postmaster if the originator address is illegal
errwarnpost	Send copies of warnings to the postmaster if the originator address is illegal
expandchannel	Channel in which to perform deferred expansion due to application of expandlimit
expandlimit	Process an incoming message "off-line" when the number of addressees exceeds this limit
expirysource	(New in JES MS 7.0) Source channel supports Expiry-date: header value
explicitaslexternal	(New in MS 8.0) Disable automatic use of AUTH EXTERNAL at MAIL FROM
expnallow	(New in JES MS 6.1) Explicitly enable support of the SMTP command EXPN
expndefault	(New in JES MS 6.1) Default handling of the SMTP command EXPN
expndisable	(New in JES MS 6.1) Disable support of the SMTP command EXPN
exproute	Explicit routing for this channel's addresses
exquota	On OpenVMS, use EXQUOTA privileges if necessary to deliver VMS MAIL messages; on UNIX treat as <code>holdexquota</code> for Berkeley mailboxes; on all platforms deliver to overquota PMDF MessageStore or PMDF popstore accounts
externalidentity	(New in MS 7.0.5) Identity for SMTP channel's client use of SMTP AUTH EXTERNAL

fileinto	Specify effect on address when a Sieve filter fileinto action is applied
filesperjob	Number of queue entries to be processed by a single job
filter	Specify the location of user Sieve filter files
fixsyntaxerrors	(New in MS 8.0) Correct syntax errors in header fields
flagtransfer	Support private XDFLG SMTP extension; pass along delivery flags, such as trusting a subaddress
foreign	DEPRECATED: Use VMS MAIL's foreign message format as needed with VMS MAIL
forwardcheckdelete	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, delete the name and use the IP address
forwardchecknone	Do not perform a forward lookup after a DNS reverse lookup
forwardchecktag	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, tag the name with *
futurerelease	(New in MS 7.0) Enable source channel support for the future release SMTP extension
generatemessagehash	(New in JES MS 6.3) Generate a hash of specified message header fields
goldmail	DEPRECATED: Generate Gold-Mail compatible read receipts
header_733	Use % routing in the message header
header_822	Use source routes in the message header
header_uucp	Use ! routing in the header
headerbottom	DEPRECATED: Place the message header at the bottom of the message (usage discouraged; use with caution)
headerfoldpreserve	Attempt to preserve original header line fold points
headerfoldremove	Re-do header line folding
headerinc	Place the message header at the top of the message
headerkeeporder	(New in MS 6.3) Preserve ordering of header lines
headerlabelalignment	Align headers
headerlimit	Truncate message header at specified size
headerlineincrement	Increment used when attempting to fold header lines
headerlinelength	Fold long headers
headeromit	DEPRECATED: Omit the message header from the message (usage discouraged; use with caution)
headerread	Apply source channel header trimming rules from a header option file to the message headers before headers are processed (use with caution)
headerset7	RESTRICTED: Decode the specified charset in headers when dequeuing
headerset8	RESTRICTED: Decode the specified charset in headers when dequeuing
headersetesc	RESTRICTED: Decode the specified charset in headers when dequeuing
headertrailingpreserve	Preserve trailing spaces (including before fold points) in header lines
headertrailingremove	Remove trailing spaces (including before fold points) in header lines
headertrim	Apply destination channel header trimming rules from a header option file to the message headers after headers are processed (use with caution)
holdexquota	Hold messages for users that are over quota
holdlimit	.HELD an incoming message when the number of addressees exceeds this limit
identnone	Do not perform IDENT lookups; do perform IP to hostname translation; include both hostname and IP address in Received: header
identnonelimited	Do not perform IDENT lookups; do perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
identnonenumeric	Do not perform IDENT lookups or IP to hostname translation
identnonesymbolic	Do not perform IDENT lookups; do perform IP to hostname translation; include only the hostname in Received: header
identtcp	Perform IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header
identtcplimited	Do perform IDENT lookups; do perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
identtcpnumeric	Perform IDENT lookups on incoming SMTP connections, but do not perform IP to hostname translation
identtcpsymbolic	Perform IDENT lookups on incoming SMTP connections and IP to hostname translation; include only hostname in Received: header
ignoreencoding	Ignore Encoding: header on incoming messages
ignoremessageencoding	(New MS 6.3.) Ignore any Content-transfer-encoding: header line present on incoming MIME message parts
ignoremultipartencoding	(New in MS 6.3.) Ignore any Content-transfer-encoding: header line present on incoming MIME multipart parts
immediate	Delivery started immediately after submission for messages of appropriate priority
immmnonurgent	Delivery started immediately after submission even for messages with lower than normal priority
immmnormal	Delivery started immediately after submission for messages of normal or higher priority
immmurgent	Delivery started immediately after submission for urgent messages only

Alphabetic list of channel options

implicitsaslexternal	(New in MS 8.0) Enable automatic use of AUTH EXTERNAL at MAIL FROM after successful negotiation of TLS
improute	Implicit routing for this channel's addresses
includefinal	Include final form of address in delivery notifications
inner	Rewrite inner message headers
innertrim	Apply header trimming rules from an options file to inner message headers (use with caution)
interfaceaddress	Bind to the specified TCP/IP interface address
interpretencoding	Interpret Encoding; header on incoming messages
interpretmessageencoding	(New in MS 6.3.) Interpret any Content-transfer-encoding; header line present on incoming MIME message parts
interpretmultipartencoding	(New in MS 6.3.) Interpret any Content-transfer-encoding; header line present on incoming MIME multipart parts
keepmessagehash	(New in JES MS 6.3) Keep message hash
language	Specify a preferred language for messages that have none
lastresort	Specify a last resort host
limitheadertermination	(New in MS 7.0.5) Only CRLF CRLF terminates message header
linelength	Message lines exceeding this length limit will be wrapped
linelimit	Maximum number of lines allowed per message
lmtp	Channel uses LMTP
lmtp_cr	Accept CR as an LMTP line terminator
lmtp_crlf	Require CRLF as the LMTP line terminator
lmtp_crorlf	Allow any of CR, LF, or CRLF as the LMTP line terminator
lmtp_lf	Accept LF as an LMTP line terminator
localbehavior	RESTRICTED: Control some local-channel-like behaviors
localvrfy	Issue SMTP VRFY command using local address
logging	Log message enqueues and dequeues into the log file
logheader	Include message headers in the message log entries
loopcheck	Enable support for the XLOOP SMTP extension (used to detect a type of message loop)
mailfromdnsverify	Verify that the domain specified on MAIL FROM: line is in the DNS
master	Channel is served only by a master program
master_debug	Generate debugging output in the channel's master program output
maxblocks	Maximum number of MTA blocks per message; longer messages are broken into multiple messages
maxheaderaddrs	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines
maxheaderchars	Maximum number of characters per message header line; longer header lines are broken into multiple header lines
maxjobs	Maximum number of jobs which can be created at once
maxlines	Maximum number of message lines per message; longer messages are broken into multiple messages
maxperiodicnonurgent	Specify that periodic jobs should only process messages of non-urgent or lower priority
maxperiodicnormal	Specify that periodic jobs should only process messages of normal or lower priority
maxperiodicurgent	Specify that periodic jobs should process messages of urgent or lower priority
maxprocchars	Specify maximum length of headers to process
maysasl	Allow SMTP server and client SASL authentication
maysaslclient	SMTP client attempts to use SASL authentication
maysaslserver	SMTP server offers SASL authentication
maytls	SMTP client and server allow TLS use
maytlsclient	SMTP client will attempt TLS use
maytlsserver	SMTP server allows TLS use
minperiodicnonurgent	Specify that periodic jobs should only process messages of non-urgent or higher priority
minperiodicnormal	Specify that periodic jobs should only process messages of normal or higher priority
minperiodicurgent	Specify that periodic jobs should only process messages of urgent priority
missingrecipientpolicy	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
mlslabel	(New in MS 7.0) RESTRICTED: Not yet fully implemented
mlsrange	(New in MS 7.0) RESTRICTED: Not yet fully implemented
msexchange	Channel serves MS Exchange gateways
mtprioritiesallowed	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; other values adjusted to be within range
mtprioritiesrequired	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; messages with other values will be rejected.
multigate	Channel serves multiple BITNET gateways, or LMTP back ends
multiple	Accepts multiple destination hosts in a single message copy

mustsasl	Must use SASL authentication
mustsasliclient	SMTP client insists upon SASL authentication
mustsaslserver	SMTP server insists upon SASL authentication
musttls	SMTP client and server insist upon TLS use and will not transfer messages with remote sides that do not support TLS
musttlsclient	SMTP client insists upon TLS use and will not send messages to any remote SMTP server that does not support TLS use
musttlsserver	SMTP server insists upon TLS use and will not accept messages from any remote SMTP client that does not support TLS use
mx	TCP/IP network and software supports MX record lookups
nameparameterlengthlimit	Maximum length to allow for NAME parameter of MIME Content-type: header line
nameservers	Consult specified nameservers rather than TCP/IP stack's choice (only for MX records as of MS 7.0)
noaddlineaddr	Do not attempt the (risky) extraction of recipient addresses from VMS MAIL header lines
noaddressrs	(New in MS 6.3p1) Addresses matching this channel will not be SRS encoded.
noaddrreturnpath	Do not add a Return-Path: header line
noaddrtypescan	(New in MS 7.0.5) Do not store recipient address "type" in an envelope flag
nobangoverpercent	Group A!B%C as (A!B)%C (default)
nobinaryclient	(New in MS 6.3.) Disable BINARYMIME support in the SMTP client
nobinaryserver	(New in MS 6.3.) Disable BINARYMIME support in the SMTP server
noblocklimit	No limit specified for the number of MTA blocks allowed per message
nocache	Do not cache any connection information
nochannelfilter	Do not perform channel filtering for outgoing messages; synonym for nodeestinationfilter
nochunkingclient	(New in MS 6.3) Disable CHUNKING support in the SMTP client
nochunkingserver	(New in MS 6.3) Disable CHUNKING support in the SMTP server
noconvert_octet_stream	Alias for noconverttooctetstream ; in legacy configuration, do not convert application/octet-stream material
noconverttooctetstream	(Unified Configuration only) Do not convert application/octet-stream material
nodayofweek	Remove day of week from date/time specifications
nodefaulthost	Do not specify a domain name to use to complete addresses
nodeferred	Alias for nodeferreddestination (do not honor deferred delivery dates)
nodeferreddestination	(New in MS 7.0u4) Do not honor deferred delivery dates
nodeferredsource	(New in MS 7.0u4) Do not honor deferred delivery dates
nodefragment	Do not perform special processing for message/partial messages
nodeestinationfilter	Do not perform channel filtering for outgoing messages
nodestinationsrs	(New in MS 6.3p1) Messages matching this destination channel will not have addresses SRS encoded .
nodns	TCP/IP network does not support DNS (nameserver) lookups; on UNIX, merely disables MX lookups since on UNIX it is <code>nsswitch.conf</code> that controls consultation of nameservers
nodropblank	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
noehlo	Never use the SMTP EHLO command
noexpirysource	(New in JES MS 7.0) Source channel ignores Expiry-date: header value
noexproute	No explicit routing for this channel's addresses
noexquota	Return to originator any messages to users who are over quota
nofileinto	Sieve filter fileinto action has no effect
nofilter	No external-to-LDAP user Sieve filters
noflagtransfer	Disable XDFLG SMTP extension; do not pass along delivery flags
noflowthrough	RESTRICTED: Not yet implemented
noforeign+	Do not use VMS MAIL's foreign message format
nogoldmail+	Do not generate Gold-Mail compatible read receipts
noheaderread	Do not apply header trimming rules from option file upon message enqueue
noheadertrim	Do not apply header trimming rules from header option file
noimproute	No implicit routing for this channel's addresses
noinner	Do not rewrite inner message headers
noinnertrim	Do not apply header trimming to inner message headers
nolinelimit	No limit specified for the number of lines allowed per message
nolocalbehavior	No special local-channel-like behavior requested
nologging	Do not log message enqueues and dequeues into the MTA message transaction log file
noloopcheck	Disable support for the XLOOP SMTP extension (used to detect a type of message loop)
nomailfromdnsverify	Do not perform DNS domain verification on the MAIL FROM: address
nomaster_debug	Do not generate debugging output in the channel's master program output

Functional group list of channel options

nomsexchange	Channel does not serve MS Exchange gateways
nomultigate	Channel does not serve multiple BITNET gateways
nomx	TCP/IP network does not support MX lookups
nonotary	RESTRICTED: Disable DSN extension use by SMTP/LMTP client
nonrandommx	Do MX lookups; do not randomize returned entries with equal precedence
nonurgentafter	Specify time delay before master channel programs run for non-urgent priority messages
nonurgentbackoff	Channel delivery retry backoff intervals for non-urgent priority messages
nonurgentblocklimit	Force messages above this size to wait unconditionally for a periodic job
nonurgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of non-urgent priority
nonurgentqueue	OBSOLETE: Job Controller manages queue priority internally; (formerly specified the queue for master channel program processing of non-urgent priority messages)
noreceivedfor	Do not include envelope To address in Received: header
noreceivedfrom	Do not include the envelope From address when constructing Received: header
noremotehost	Use local host's domain name as the default domain name to complete addresses
norestricted	Do not apply RFC 1137 restricted encoding to addresses
noreturnaddress	Use the value of the return_address MTA option
noreturnpersonal	Use the value of the return_personal MTA option
noreverse	Do not apply address reversal to addresses
normalafter	Specify time delay before master channel programs run for normal priority messages
normalbackoff	Channel delivery retry backoff intervals for normal priority messages
normalblocklimit	Force messages above this size to nonurgent priority
normalnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of normal priority
normalqueue	Specify the queue for master channel program processing of normal messages
norules	Do not do channel-specific rewrite rule checks
nosasl	SASL authentication not attempted or permitted
nosaslclient	SMTP client does not attempt SASL authentication
nosaslpassauth	Do not pass along a MAIL FROM AUTH parameter
nosaslserver	SMTP server does not permit SASL authentication
nosaslswitchchannel	Do not switch channel upon successful SASL authentication
nosasltrustauth	(New in MS 7.0u3) Do not promote a MAIL FROM AUTH parameter to the MTA's authenticated originator address envelope field
nosendetrn	Do not send SMTP ETRN command
nosendpost	Do not send copies of failures to the postmaster
noserviceall	OBSOLETE: Job Controller unaffected; (formerly meant immediate delivery jobs process only the messages they were queued to process)
noserviceconversion	(New in MS 7.0.3) Service conversions for messages coming in this channel must be enabled via the CHARSET-CONVERSION mapping table
noslave_debug	Do not generate debugging output in the channel's slave program output
nosmtp	Channel does not use SMTP
nosocks	Do not do SOCKS connections
nosourcefilter	Source channel does not have an associated Sieve filter
nosourcesrs	(New in MS 6.3p1) Messages matching this source channel will not have addresses SRS encoded .
noswitchchannel	Stay with the server channel; do not switch to the channel associated with the originating host; do not permit being switched to
sensitivitycompanyconfidential	Allow messages of any sensitivity
xclient	(New in MS 8.0) XCLIENT SMTP extension is enabled, only one group of XCLIENT commands permitted
xclientrepeat	(New in MS 8.0) XCLIENT SMTP extension is enabled, and groups of XCLIENT commands are permitted
xclientsasl	(New in MS 8.0) XCLIENT SMTP extension is enabled, and LOGIN attribute is allowed
xclientsaslrepeat	(New in MS 8.0) XCLIENT SMTP extension is enabled, LOGIN attribute is permitted, and groups of XCLIENT commands are permitted

33.3.2 Functional group list of channel options

There are a great many channel options available for configuring channel behavior. The channel options may be viewed in an [alphabetic list](#); alternatively, they may also be viewed [grouped by functionality](#) for convenience in considering inter-related options.

In Unified Configuration, channel options are configured directly under the channel name:

```
channel:channel-name.channel-option
```

In legacy configuration, channel options were set as keywords on the channel definition in the MTA configuration file, `imta.cnf`; channel options appeared on the first line of a channel definition, after the channel name.

Note that there are also many [MTA options](#) affecting MTA operation in general (rather than affecting specific channels); in general such MTA level options are distinct from channel options (not merely global defaults for channels) as MTA options may alter more fundamental aspects of MTA operation, but in some cases an MTA level option does establish a default for all channels which may then be overridden via a channel option analogue.

Note also that besides the normal channel options under discussion here, some channels also support some channel-specific options. These are channel-specific options which are *only* supported and available for that specific type of channel: for historical (and functional) reasons they are implemented differently from the usual channel options. Such channel-specific options were, in legacy configuration, configured in channel option files; in Unified Configuration they are configured under the `options` channel option:

```
channel:channel-name.options.channel-specific-option-name
```

See the [specific type of channel](#) for a list of that channel's valid channel-specific options.

[Channel options listed alphabetically](#) lists channel options alphabetically; [Channel options grouped by functionality](#) below lists channel options by functional group. Keywords shown in bold face type are defaults; keywords marked with + are only supported under OpenVMS.

Table 33.4 Channel options grouped by functionality

Option	Usage
Addresses	
733	Use % routing in the envelope; alias for <code>percents</code>
822	Use source routes in the envelope; alias for <code>sourceroute</code>
<code>acceptalladdresses</code>	(New in JES MS 6.1) Accept messages despite certain errors that would normally cause message rejection
<code>acceptvalidaddresses</code>	(New in JES MS 6.1) Perform normal rejection checks on incoming messages
<code>addlineaddr</code>	RESTRICTED: Attempt to extract additional envelope recipient addresses from X-VMS-To: and X-VMS-Cc: header lines
<code>addresssrs</code>	(New in MS 6.3p1) Addresses matching this channel are eligible for SRS encoding
<code>addreturnpath</code>	Add a Return-Path: header line
<code>addrtypescan</code>	(New in MS 7.0.5) Store recipient address "type" in an envelope flag
<code>addrtypescanbccdefault</code>	(New in MS 7.0.5) Store recipient address "type" in an envelope flag, assuming unmatched recipient addresses are Bcc: addresses
<code>aliasdetourhost</code>	Specify an "override" <code>mailHost</code> for any user found in LDAP; effect is to "detour" messages for such a user to the specified host
<code>aliaslocal</code>	Look up <code>aliases</code> ; <i>i.e.</i> , query <code>alias file</code> and <code>alias database</code> and <code>alias options</code> and perform <code>alias_url1*</code> lookups
<code>aliasmagic</code>	(New in JES MS 6.0) RESTRICTED: Destination channel override of the <code>alias_magic</code> MTA option, controlling the order of the different types of <code>alias lookups</code>
<code>aliasoptindetourhost</code>	(New in JES MS 6.2p4) Specify an "override" <code>mailHost</code> for any user who is opted-in via whatever LDAP attribute is named by the <code>ldap_detourhost_optin</code> MTA option; the effect is to "detour" messages for such a user to the specified host
<code>aliaswild</code>	Do an * lookup if no exact alias match is found
<code>authrewrite</code>	Use SMTP AUTH information in header
<code>bangonly</code>	Source channel: disable interpretation of % host-routing
<code>bangoverpercent</code>	Group A!B%C as A!(B%C)
<code>bangstyle</code>	Use UUCP ! routing in the envelope; (<code>uucp</code> from legacy configuration is an alias for <code>bangstyle</code>)
<code>defaulthost</code>	Specify a domain name to use to complete addresses

Functional group list of channel options

dequeue_removeoute	Alias for dequeue_removeoute ; in legacy configuration, strip source route (@source-route:address) when dequeuing message
dequeue_removeoute	(Unified Configuration only) Strip source route (@source-route:address) when dequeuing message
destinationrs	(New in MS 6.3p1) Messages destined out this channel are eligible for SRS encoding
enqueue_removeoute	Alias for enqueue_removeoute ; in legacy configuration, strip source route (@source-route:address) when enqueueing message
enqueue_removeoute	(Unified Configuration only) Strip source route (@source-route:address) when enqueueing message
exproute	Explicit routing for this channel's addresses
holdlimit	.HELD an incoming message when the number of addressees exceeds this limit
improute	Implicit routing for this channel's addresses
localbehavior	RESTRICTED: Control some local-channel-like behavior
missingrecipientpolicy	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
noaddlineaddr	Do not attempt the (risky) extraction of recipient addresses from VMS MAIL header lines
noaddressrs	(New in MS 6.3p1) Addresses matching this channel will not be SRS encoded.
noaddrreturnpath	Do not add a Return-Path: header line
noaddrtypescan	Do not store recipient address "type" in an envelope flag
nobangoverpercent	Group A!B%C as (A!B)%C (default)
nodefaulthost	Do not specify a domain name to use to complete addresses
nolocationrs	(New in MS 6.3p1) Messages matching this destination channel will not have addresses SRS encoded.
noexproute	No explicit routing for this channel's addresses
noimproute	No implicit routing for this channel's addresses
nolocalbehavior	No special local-channel-like behavior requested
noremotehost	Use local host's domain name as the default domain name to complete addresses
norestricted	Do not apply RFC 1137 restricted encoding to addresses
noreverse	Do not apply address reversal to addresses
norules	Do not do channel-specific rewrite rule checks
nosourcesrs	(New in MS 6.3p1) Messages matching this source channel will not have addresses SRS encoded.
percents	Use % routing in the envelope; (legacy configuration's 733 is an alias for percents)
recipientcutoff	Set a limit for the number of recipients-per-message-copy accepted on a channel; attempts to submit a message with more than this number of recipients will cause <i>all</i> recipients to be rejected
recipientlimit	Set a limit for the number of recipients-per-message-copy accepted on a channel; additional recipients will get a temporary rejection error
remotehost	Use remote host's name as the default domain name to complete addresses
restricted	Apply RFC 1137 restricted encoding to addresses
reverse	Apply address reversal to addresses in messages destined to this channel; <i>i.e.</i> , apply reverse_url LDAP-provisioned address reversal, the reverse database , and the REVERSE mapping table to addresses
routelocal	Rewriting should shortcircuit routing addresses
rules	Do channel-specific rewrite rule checks
sourceroute	Use source routes in the message envelope; (_822 is an alias for sourceroute , synonymous with legacy configuration's 822)
sourcesrs	(New in MS 6.3p1) Messages from this source channel are eligible for SRS encoding
subaddressexact	Alias must match exactly, including exact subaddress match
subaddressrelaxed	Alias without subaddress may match
subaddresswild	Alias with subaddress wildcard may match
unrestricted	Do not apply RFC 1137 restricted encoding to addresses
uucp	Alias for bangstyle
validatelocalnone	Enqueueing channels perform no validation check on the local part of addresses they enqueue to this channel
validatelocalsystem	Enqueueing channels check that the local part of addresses they enqueue to this channel matches an account on the system
viaaliasoptional	Alias match not required
viaaliasrequired	Alias match required
Attachments and MIME processing	
conditionalsecuritymultipart	Process inside security multipart, retaining preamble material
convert_octet_stream	Alias for convertoctetstream ; in legacy configuration, convert application/octet-stream material as appropriate
convertoctetstream	(Unified Configuration only) Convert application/octet-stream material as appropriate
defragment	Reassemble any MIME-compliant message/partial parts queued to this channel
foreign+	(OpenVMS only) Use VMS MAIL's foreign message format as needed with VMS MAIL
ignoreencoding	Ignore Encoding: header on incoming messages
ignoremessageencoding	(New in MS 6.3.) Ignore any Content-transfer-encoding: header line present (illegally) on incoming MIME message parts

Functional group list of channel options

ignoremultipartencoding	(New in MS 6.3.) Ignore any Content-transfer-encoding: header line present on incoming MIME multipart parts
interpretencoding	Interpret Encoding: header on incoming messages
interpretmessageencoding	(New in MS 6.3.) Interpret any Content-transfer-encoding: header line present (illegally) on incoming MIME message parts
interpretmultipartencoding	(New in MS 6.3.) Interpret any Content-transfer-encoding: header line present on incoming MIME multipart parts
linelength	Message lines exceeding this length limit will be wrapped
maxblocks	Maximum number of MTA blocks per message; longer messages are broken into multiple messages
maxlines	Maximum number of message lines per message; longer messages are broken into multiple messages
nameparameterlengthlimit	Maximum length to allow for NAME parameter of MIME Content-type: header line
noconvert_octet_stream	Alias for noconvertoctetstream ; in legacy configuration, do not convert application/octet-stream material
noconvertoctetstream	(Unified Configuration only) Do not convert application/octet-stream material
nodefragment	Do not perform special processing for message/partial messages
noforeign+	(OpenVMS only) Do not use VMS MAIL's foreign message format
nolinelimit	No limit specified for the number of lines allowed per message
nothurman	Do not perform thurman format conversion
nouma	Do not perform "thurman on demand" format conversion
parameterformatdefault	(New in MS 7.0.5) Normal handling of MIME parameters, using RFC 2231 encoding when appropriate
parameterformatminimizeencoded	(New in MS 7.0.5) Remove unnecessary, redundant RFC 2231 encoding from MIME parameters
parameterformatstripencoded	(New in MS 7.0.5) Strip any characters requiring RFC 2231 encoding from MIME parameters, so that RFC 2231 encoding may be removed
parameterlengthlimit	Maximum length to allow for parameters of MIME Content-type: header line
passthrough	Do no message processing
processsecuritymultiparts	Process inside security multiparts
retainsecuritymultiparts	Do not process inside security multiparts
thurman	Convert uuencoded and Binhex "blobs" to MIME format
uma	Perform thurman conversion if would be MIME-ifying the message anyway for other reasons
BSMTP and Bitnet	
contchar	DEPRECATED: Specify batch SMTP continuation line character
contposition	DEPRECATED: Specify folding point in batch SMTP lines
localuser	DEPRECATED: Use local usernames as user tags
nolocaluser	DEPRECATED: Do not use local usernames as user tags
notick	RESTRICTED: Do not put a ticket in the BSMTP stream
tick	RESTRICTED: Put a ticket in the BSMTP stream
verb_never	RESTRICTED: Ignore VERB commands in the BSMTP stream
verb_none	RESTRICTED: Accept VERB commands in the BSMTP stream
verb_off	RESTRICTED: Insert a VERB OFF command into the BSMTP stream
verb_on	RESTRICTED: Insert a VERB ON command into the BSMTP stream
Character sets and eight bit data	
charset7	Default character set to associate with 7-bit text messages
charset8	Default character set to associate with 8-bit text messages
charsetesc	Default character set to associate with 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 that contain unnegotiated eight bit data
headerset7	RESTRICTED: Decode the specified charset in headers when dequeuing
headerset8	RESTRICTED: Decode the specified charset in headers when dequeuing
headersetesc	RESTRICTED: Decode the specified charset in headers when dequeuing
parameterformatdefault	(New in MS 7.0.5) Normal handling of MIME parameters, using RFC 2231 encoding when appropriate
parameterformatminimizeencoded	(New in MS 7.0.5) Remove unnecessary, redundant RFC 2231 encoding from MIME parameters
parameterformatstripencoded	(New in MS 7.0.5) Strip any characters requiring RFC 2231 encoding from MIME parameters, so that RFC 2231 encoding may be removed
sevenbit	Channel does not support eight bit characters; eight bit characters must be encoded
Conversion tags and service conversions	
destinationconversiontag	(New in MS 7.0.5) Conversion tags to add to outgoing recipients
noserviceconversion	(New in MS 7.0.3) Service conversions for messages coming in this channel must be enabled via the CHARSET-CONVERSION mapping table

Functional group list of channel options

serviceconversion	(New in MS 7.0.3) Perform service conversions for messages coming in this channel regardless of CHARSET-CONVERSION
sourceconversiontag	(New in MS 7.0.5) Conversion tags to add to message
Display labels	
caption	(New in JES MS 6.3) Channel caption: a short description, suitable as the caption for a column of a table
description	Channel description
DKIM	
dkimignore	(New in MS 7.0.5) Take no special action in regards to DKIM-Signature: header fields
dkimpreserve	(New in MS 7.0.5) Don't rewrite DKIM-signed messages for specified domains
dkimremove	(New in MS 7.0.5) Remove DKIM signatures
Error interpretation	
usepermanenterror	(New in MS 8.0) Override use_permanent_error MTA option on a per-source-channel basis
usetemporaryerror	(New in MS 8.0) Override use_temporary_error MTA option on a per-source-channel basis
File creation in the MTA queue area	
addrspersfile	Number of addresses per message file
expandchannel	Channel in which to perform deferred expansion due to application of expandlimit
expandlimit	Process an incoming message "off-line" when the number of addressees exceeds this limit
multiple	Accepts multiple destination hosts in a single message copy
single	Only one envelope To address per message copy
single_sys	Each message copy must be for a single destination system
subdirs	Use multiple subdirectories for messages queued to channel
Gateway/firewall/mailhub/Message Router channel connection	
aliasdetourhost	Specify an "override" mailHost for any user found in LDAP; effect is to "detour" messages for such a user to the specified host
aliasoptindetourhost	(New in JES MS 6.2p4) Specify an "override" mailHost for any user who is opted-in via whatever LDAP attribute is named by the ldap_detourhost_optin MTA option; the effect is to "detour" messages for such a user to the specified host
daemon	Specify name of a gateway daemon (host) to route to
lastresort	Specify a last resort host
multigate	Channel serves multiple BITNET gateways, or multiple LMTP back ends
nomultigate	Channel does not serve multiple BITNET gateways, or multiple LMTP back ends
user	DEPRECATED as of MS 8.0; instead use the pipeuser option in restricted.cnf . Specify account under which to run the pipe channel or specify Message Router mailbox name
Headers	
addrreturnpath	Add a Return-Path: header line
authrewrite	Use SMTP AUTH information in header
commentinc	Leave comments in message header lines intact
commentmap	Apply COMMENT_STRINGS mapping to comments in message header lines
commentomit	Remove comments from message header lines
commentstrip	Remove problematic characters from comment field in message header lines
commenttotal	Strip comments (material in parentheses) everywhere
datefour	Convert date/time specifications to four digit years
datetwo	Convert date/time specifications to two digit years
dayofweek	Include day of week in date/time specifications
defaulthost	Specify a domain name to use to complete addresses
dropblank	Strip blank To, Resent-To, Cc., or Resent-Cc: headers
fixsyntaxerrors	(New in MS 8.0) Correct syntax errors in header fields
header_733	Use % routing in the message header
header_822	Use source routes in the message header
header_uucp	Use ! routing in the header
headerbottom+	(OpenVMS only) Place the message header at the bottom of the message (usage discouraged; use with caution)
headerfoldpreserve	Attempt to preserve original header line fold points
headerfoldremove	Re-do header line folding
headerinc	Place the message header at the top of the message; <i>i.e.</i> , normal handling
headerkeeporder	(New in MS 6.3) Preserve ordering of header lines
headerlabelalignment	Align headers
headerlimit	Truncate message header at specified size
headerlineincrement	Increment used when attempting to fold header lines

Functional group list of channel options

headerlinelength	Fold long headers
headeromit+	(OpenVMS only) Omit the message header from the message (usage discouraged; use with caution)
headerread	Apply source channel header trimming rules from a header option file to the message headers before headers are processed (use with caution)
headerset7	RESTRICTED: Decode the specified charset in headers when dequeuing
headerset8	RESTRICTED: Decode the specified charset in headers when dequeuing
headersetesc	RESTRICTED: Decode the specified charset in headers when dequeuing
headertrim	Apply destination channel header trimming rules from an header option file to the message headers after headers are processed (use with caution)
inner	Rewrite inner message headers
innertrim	Apply header trimming rules from a header option file to inner message headers (use with caution)
limithadertermination	(New in MS 7.0.5) Only CRLF terminates message header
maxheaderadrs	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines
maxheaderchars	Maximum number of characters per message header line; longer header lines are broken into multiple header lines
missingrecipientpolicy	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
noaddrreturnpath	Do not add a Return-Path: header line
nodayofweek	Remove day of week from date/time specifications
nodefaulthost	Do not specify a domain name to use to complete addresses
nodropblank	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
noheaderread	Do not apply header trimming rules from any header option file upon message enqueue
noheadertrim	Do not apply header trimming rules from any header option file
noinner	Do not rewrite inner message headers
noinnertrim	Do not apply header trimming to inner message headers
noreceivedfor	Do not include envelope To address in Received: header
noreceivedfrom	Do not include the envelope From address when constructing Received: header
noremotehost	Use local host's domain name as the default domain name to complete addresses
norestricted	Do not apply RFC 1137 restricted encoding to addresses
noreverse	Do not apply address reversal to addresses
norules	Do not do channel-specific rewrite rule checks
nox_env_to	Do not add X-Envelope-to: header lines while enqueueing
passyntaxerrors	(New in MS 8.0) Disable certain header file syntax error fixup
passthrough	Do no message processing
personalinc	Leave personal names in message header lines intact
personalmap	Apply PERSONAL_NAMES mapping to personal names in message header lines
personalomit	Remove personal name fields from message header lines
personalstrip	Strip problematic characters from personal name fields in message header lines
receivedfor	Include envelope to address in Received: header
receivedfrom	Include the envelope From address when constructing Received: header
receivedstate	(New in MS 8.0) Include a state indicator when constructing Received: header
relaxheadertermination	Allow lines containing white space characters to terminate the header of a message
remotehost	Use remote host's name as the default domain name to complete addresses
restricted	Apply RFC 1137 restricted encoding to addresses
reverse	Apply address reversal to addresses; that is, apply reverse_url LDAP-based address reversal, the reverse database , and the REVERSE mapping to addresses
rules	Do channel-specific rewrite rule checks
sensitivitycompanyconfidential	Allow messages of any sensitivity
sensitivitynormal	Reject messages whose sensitivity is higher than normal
sensitivitypersonal	Reject messages whose sensitivity is higher than personal
sensitivityprivate	Reject messages whose sensitivity is higher than private
sourcecommentinc	Leave comments in incoming message header lines intact
sourcecommentmap	Apply COMMENT_STRINGS mapping to comments in incoming message header lines
sourcecommentomit	Remove comments from incoming message header lines
sourcecommentstrip	Remove problematic characters from comment field in incoming message header lines
sourcecommenttotal	Strip comments (material in parentheses) everywhere in incoming messages
sourcepersonalinc	Leave personal names in incoming message header lines intact

Functional group list of channel options

sourcepersonalmap	Apply PERSONAL_NAMES mapping to personal names in incoming message header lines
sourcepersonalomit	Remove personal name fields from incoming message header lines
sourcepersonalstrip	Strip problematic characters from personal name fields in incoming message header lines
unrestricted	Do not apply RFC 1137 restricted encoding to addresses
usereplyto	Specify mapping of Reply-to: header
useresent	Specify mapping of Resent- headers for non RFC 822 environments
x_env_to	Add X-Envelope-to: header lines while enqueueing
Host names	
additional_host_names	(Unified Configuration only) Additional hosts the channel can reach
daemon	Specify name of a gateway daemon to route to
defaulthost	Specify a domain name to use to complete addresses
lastresort	Specify a last resort host
local_host_alias	(Unified Configuration only) Override official_host_name on outgoing messages
nodefaulthost	Do not specify a domain name to use to complete addresses
noremotehost	Use local host's domain name as the default domain name to complete addresses
official_host_name	(Unified Configuration only) Official host name for the channel
remotehost	Use remote host's name as the default domain name to complete addresses
Incoming channel matching and switching	
allowswitchchannel	Allow switching to this channel from a switchchannel channel
nosaslswitchchannel	Do not switch channel upon successful SASL authentication
noswitchchannel	Stay with the server channel; do not switch to the channel associated with the originating host; do not permit being switched to
saslswitchchannel	Switch to another channel when SASL authentication is successful; the channel to switch to is specified via the value of the user's <code>mailSMTPSubmitChannel</code> LDAP attribute (or as of MS 8.0, whatever LDAP attribute is named by the <code>ldap_auth_attr_submit_channel</code> MTA option)
switchchannel	Switch effective source channel to the channel associated with the originating host's source IP
tlsswitchchannel	Switch to specified channel upon successful TLS negotiation
userswitchchannel	(New in MS 6.3) Enable switching from this source channel to a channel selected via a per-user or per-domain LDAP attribute
Logging and debugging	
logging	Log message enqueues and dequeues into the MTA message transaction log file
logheader	Include message header records in the MTA message transaction log file
master_debug	Generate debugging output in the channel's master program output
nologging	Do not log message enqueues and dequeues into the MTA message transaction log file
nomaster_debug	Do not generate debugging output in the channel's master program output
noslave_debug	Do not generate debugging output in the channel's slave program output
slave_debug	Generate debugging output in the channel's slave program output
Long address lists or headers	
alternatchannel	Messages that exceed a channel's alternatblocklimit , alternatelinelimit , or alternaterecipientlimit will be diverted to the channel's specified alternatchannel
alternaterecipientlimit	Divert messages that exceed the specified number of recipients to the alternatchannel
deferralrejectlimit	(New in MS 6.2) Limit the number of bad (failing) recipient addresses
disconnectrecipientlimit	(New in MS 6.2) Force SMTP session disconnect after the specified number of recipients is exceeded
disconnectrejectlimit	(New in MS 6.2) Force SMTP session disconnect after the specified number of bad recipients (rejected recipients) is exceeded
expandchannel	Channel in which to perform deferred expansion due to application of expandlimit
expandlimit	Process an incoming message "off-line" when the number of addressees exceeds this limit
holdlimit	.HELD an incoming message when the number of addressees exceeds this limit
maxprocchars	Specify maximum length of headers to process
recipientcutoff	Set a limit for the number of recipients-per-message-copy accepted on a channel; attempts to submit a message with more than this number of recipients will cause <i>all</i> recipients to be rejected
recipientlimit	Set a limit for the number of recipients-per-message-copy accepted on a channel; additional recipients will get a temporary rejection error
Message hashes	
deletemessagehash	(New in MS 6.3) Delete message hash
generatemessagehash	(New in MS 6.3) Generate a hash of specified message header fields
keepmessagehash	(New in MS 6.3) Keep message hash
Message tracking	
notrackingclient	(New in MS 8.0.) Disable SMTP client support of message tracking

notrackingserver	(New in MS 8.0.) Disable SMTP server support of message tracking
trackingclient	(New in MS 8.0.) Enable SMTP client support of message tracking
trackingdelivered	(New in MS 8.0.) Treat messages dequeued from this channel as delivered
trackingfirst	(New in MS 8.0.) Tracking information goes to first alias expansion result
trackinggenerate	(New in MS 8.0.) Enable SMTP client support of message tracking
trackinginternal	(New in MS 8.0.) This channel transfers internally
trackingmultiple	(New in MS 8.0.) Tracking information passes through aliases
trackingrelayed	(New in MS 8.0.) Treat messages dequeued from this channel as relayed
trackingserver	(New in MS 8.0.) Enable SMTP server support of message tracking
trackingsingle	(New in MS 8.0.) Only pass tracking info through single recipient alias
trackingtimeoutdefault	(New in MS 8.0.) Default timeout (s) for tracking requests
trackingtimeoutmax	(New in MS 8.0.) Maximum timeout (s) for tracking requests
trackingtimeoutmin	(New in MS 8.0.) Minimum timeout (s) for tracking requests
Multi Level Security	
mlslabel	(New in MS 7.0) RESTRICTED: Not yet fully implemented
mlsrange	(New in MS 7.0) RESTRICTED: Not yet fully implemented
Notification messages and postmaster messages	
aliaspostmaster	Redirect postmaster messages to the local channel postmaster
copysendpost	Send copies of failures to the postmaster unless the originator address is blank
copywarnpost	Send copies of warnings to the postmaster unless the originator address is blank
dispositionchannel	(New in MS 6.2) Channel to use when generating dispositions
errsendpost	Send copies of failures to the postmaster if the originator address is illegal
errwarnpost	Send copies of warnings to the postmaster if the originator address is illegal
expirysource	(New in JES MS 7.0) Source channel supports Expiry-date: header value
goldmail+	(OpenVMS only) Generate Gold-Mail compatible read receipts
includefinal	Include final form of address in delivery notifications
language	Specify a preferred language for messages that have none
noexpirysource	(New in JES MS 7.0) Source channel ignores Expiry-date: header value
nogoldmail+	Do not generate Gold-Mail compatible read receipts
nonotary	RESTRICTED: Disable DSN extension use by SMTP/LMTP client
nonurgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of non-urgent priority
noreturnaddress	Use the value of the return_address MTA option
noreturnpersonal	Use the value of the return_personal MTA option
normalnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of normal priority
nosendpost	Do not send copies of failures to the postmaster
notary	Support SMTP DSN extensions
notices	Specify the amount of time which may elapse before notices are sent and messages returned
notificationchannel	(New in MS 6.2) Channel to use when generating notifications
nowarnpost	Do not send copies of warnings to the postmaster
postheadbody	Both the message's header and body are sent to the postmaster when a delivery failure occurs
postheadonly	Only the message's header is sent to the postmaster when a delivery failure occurs
processsecuritymultiparts	Process inside security multiparts
readreceiptmail	Ignore read receipt requests when delivering to VMS MAIL, rather than "downgrading" them to delivery receipt requests; leave it up to user agents to act upon the read receipt request
reportboth	Generate both header and NOTARY delivery receipt requests from "foreign" delivery receipt requests
reporthead	Generate only header delivery receipt requests from "foreign" delivery receipt requests
reportnotary	Generate only NOTARY delivery receipt requests from "foreign" delivery receipt requests
reportsuppress	Suppress delivery receipt requests from "foreign" delivery receipt requests
retainsecuritymultiparts	Do not process inside security multiparts
returnaddress	Set the return address for the local Postmaster
returnenvelope	Control use of blank envelope return addresses
returnpersonal	Set the personal name for the local Postmaster
sendpost	Send copies of failures to the postmaster
suppressfinal	Include only original form of address in notification messages
urgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority

Functional group list of channel options

useintermediate	Include the "intermediate" form of recipient address, that form previously presented as active from the MTA's point of view, in delivery status notification messages
warnpost	Send copies of warnings to the postmaster
Processing control and job submission	
addrsperjob	Number of addresses to be processed by a single job
after	Specify time delay before master channel programs run
backoff	Channel delivery retry backoff intervals
bidirectional	Channel is served by both a master and slave program
deferred	DEPRECATED as of MS 7.0, in favor of deferreddestination , which it is an alias for in Unified Configuration; honor deferred delivery dates in Deferred-delivery: header lines;
deferreddestination	(New in MS 7.0) Honor deferred delivery dates in Deferred-delivery: header lines; (deferred is an alias for deferreddestination)
deferredsource	(New in MS 7.0) Honor deferred delivery dates in Deferred-delivery: header lines
expandchannel	Channel in which to perform deferred expansion due to application of expandlimit
expandlimit	Process an incoming message "off-line" when the number of addressees exceeds this limit
filesperjob	Number of queue entries to be processed by a single job
futurerelease	(New in MS 7.0) Enable source channel support for the future release SMTP extension
immediate	OBSOLETE: Job Controller behavior is unaffected by this option; (used to mean delivery started immediately after submission for messages of appropriate priority)
immonurgent	OBSOLETE: Job Controller behavior is unaffected by this option (though this is essentially the Job Controller behavior); (used to mean delivery started immediately after submission even for messages with lower than normal priority)
imnormal	OBSOLETE: Job Controller behavior is unaffected by this option; (used to mean delivery started immediately after submission for messages of normal or higher priority)
immurgent	OBSOLETE: Job Controller behavior is unaffected by this option; (used to mean delivery started immediately after submission for urgent messages only)
master	Channel is served only by a master program
maxjobs	Maximum number of jobs which can be created at once
maxperiodicnonurgent	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should only process messages of non-urgent or lower priority); see instead Job Controller priority-based processing
maxperiodicnormal	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should only process messages of normal or lower priority); see instead Job Controller priority-based processing
maxperiodicurgent	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should process messages of urgent or lower priority); see instead Job Controller priority-based processing
minperiodicnonurgent	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should only process messages of non-urgent or higher priority); see instead Job Controller priority-based processing
minperiodicnormal	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should only process messages of normal or higher priority); see instead Job Controller priority-based processing
minperiodicurgent	OBSOLETE: Job Controller behavior is unaffected by this option; (used to specify that periodic jobs should only process messages of urgent priority); see instead Job Controller priority-based processing
mtprioritiesallowed	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; other values adjusted to be within range
mtprioritiesrequired	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; messages with other values will be rejected.
nodeferred	Do not honor deferred delivery dates
nodeferreddestination	(New in MS 7.0u4) Do not honor deferred delivery dates
nodeferredsource	(New in MS 7.0u4) Do not honor deferred delivery dates
nonurgentafter	Specify time delay before master channel programs run for non-urgent priority messages
nonurgentbackoff	Channel delivery retry backoff intervals for non-urgent priority messages
nonurgentblocklimit	Force messages above this size to wait unconditionally for a periodic job
nonurgentqueue	OBSOLETE: Job Controller manages queue priority internally; (formerly specified the queue for master channel program processing of non-urgent priority messages)
normalafter	Specify time delay before master channel programs run for normal priority messages
normalbackoff	Channel delivery retry backoff intervals for normal priority messages
normalblocklimit	Force messages above this size to non-urgent priority
normalqueue	OBSOLETE: Job Controller manages queue priority internally; (formerly specified the queue for master channel program processing of normal priority messages)
noserviceall	OBSOLETE: Job Controller unaffected; (formerly meant immediate delivery jobs process only the messages they were queued to process)
period	OBSOLETE: Job Controller unaffected; (formerly specified periodicity of periodic channel service)
periodic	OBSOLETE: Job Controller unaffected; (formerly meant channel is serviced only periodically; immediate delivery processing is never done)
pool	Specify Job Controller pool in which channel programs run
queue	DEPRECATED: use pool instead; (specify queue/pool master channel programs run in)

Functional group list of channel options

secondclassafter	Specify time delay before channel runs for secondclass priority messages
secondclassblocklimit	Force messages above this size to third class priority
secondclassqueue	RESTRICTED: Not implemented for Messaging Server MTA; (for PMDF, specified the queue for master channel program processing of second class messages)
serviceall	OBSOLETE: Job Controller is unaffected; (formerly meant immediate delivery jobs process all messages queued for the channel)
slave	Channel is serviced only by a slave program
threaddepth	Number of messages triggering new thread with SMTP client
urgentafter	Specify time delay before master channel programs run for urgent priority messages
urgentbackoff	Channel delivery retry backoff intervals for urgent priority messages
urgentblocklimit	Force messages above this size to normal priority
urgentqueue	OBSOLETE: Job Controller manages queue priority internally; (formerly specified the queue for master channel program processing of urgent messages)
user	Specify account under which to run the pipe channel or specify Message Router mailbox name
Sensitivity limits	
sensitivitycompanyconfidential	Allow messages of any sensitivity
sensitivitynormal	Reject messages whose sensitivity is higher than normal
sensitivitypersonal	Reject messages whose sensitivity is higher than personal
sensitivityprivate	Reject messages whose sensitivity is higher than private
Sieve filters and delivery flags	
addrtypescan	(New in MS 7.0.5) Store recipient address "type" in an envelope flag
addrtypescanbccdefault	(New in MS 7.0.5) Store recipient address "type" in an envelope flag, assuming unmatched recipient addresses are Bcc: addresses
channelfilter	Alias for destinationfilter
deliveryflags	Set flags controlling certain delivery behaviors
destinationfilter	Specify the location of channel filter file for outgoing messages; (channelfilter is an alias for destinationfilter)
fileinto	Specify effect on address when a Sieve "fileinto" action is applied
filter	Specify the location of user filter files
flagtransfer	Support private XDFLG SMTP extension; pass along delivery flags, such as trusting a subaddress
noaddrtypescan	(New in MS 7.0.5) Do not store recipient address "type" in an envelope flag
nochannelfilter	Alias for noderestinationfilter
noderestinationfilter	Do not perform channel filtering on outgoing messages; (nochannelfilter is an alias for noderestinationfilter)
nofileinto	Sieve "fileinto" action has no effect
nofilter	No external-to-LDAP user Sieve filters
noflagtransfer	Disable XDFLG SMTP extension; do not pass along delivery flags
nosourcefilter	Source channel does not have an associated Sieve filter
scriptlimit	(New in MS 8.0) Maximum number of Sieve scripts a user can have
sievelimit	(New in MS 8.0) Maximum size of a user Sieve script
sizelimit	(New in MS 8.0) Maximum combined size of a user's Sieve scripts
sourcefilter	Specify the location of channel filter file for incoming messages
Size limits on messages	
alternateblocklimit	Divert messages that exceed the specified number of blocks to the alternatechannel
alternatechannel	Messages that exceed a channel's alternateblocklimit , alternatelinelimit , or alternaterecipientlimit will be diverted to the channel's specified alternatechannel
alternatelinelimit	Divert messages that exceed the specified number of lines to the alternatechannel
alternaterecipientlimit	Divert messages that exceed the specified number of recipients to the alternatechannel
blocklimit	Maximum number of MTA blocks allowed per message
exquota	OBSOLETE: PMDF only
holdexquota	OBSOLETE: PMDF only
holdlimit	.HELD an incoming message when the number of addressees exceeds this limit
linelimit	Maximum number of lines allowed per message
noblocklimit	No limit specified for the number of MTA blocks allowed per message
noexquota	Return to originator any messages to users who are over quota
nonurgentblocklimit	Force messages above this size to wait unconditionally for a periodic job
normalblocklimit	Force messages above this size to non-urgent priority
sourceblocklimit	Maximum number of MTA blocks allowed per incoming message
urgentblocklimit	Force messages above this size to normal priority

Functional group list of channel options

sourcespamfilter2	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued by this source channel
sourcespamfilter2optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter3	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued by this source channel
sourcespamfilter3optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter4	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued by this source channel
sourcespamfilter4optin	(New in JES MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilteroptin	Alias for sourcespamfilterloptin
SMTP and LMTP protocol	
allowetrn	Honor SMTP client ETRN commands
binaryclient	(New in MS 6.3.) RESTRICTED: Not yet fully implemented. Enable BINARYMIME support in the SMTP client
binaryserver	(New in MS 6.3 but at that time RESTRICTED: Not yet fully implemented; actual implementation new in MS 8.0) Enable BINARYMIME support in the SMTP server
blocketrn	Do not honor SMTP client ETRN commands
checkehlo	Check the SMTP greeting banner for whether to use EHLO
chunkingclient	(New in MS 6.3-0.15) Enable CHUNKING support in the SMTP client
chunkingserver	(New in MS 6.3-0.15) Enable CHUNKING support in the SMTP server
conditionalpassthrough	"Pass-through" mode, if any Received: header lines are present
conditionalrelay	"Relay" mode, if any Received: header lines are present
deferralrejectlimit	(New in MS 6.2) Limit the number of bad (failing) recipient addresses
destinationmosolicit	(New in MS 6.2) List of solicitation types not accepted
disableetrn	Disable support for the ETRN SMTP command
disconnectbadburlimit	(New in MS 7.0u4) Force SMTP SUBMIT session disconnect after a specified number of invalid BURL commands is exceeded
disconnectbadcommandlimit	New in (MS 6.2) Force SMTP session disconnect after a specified number of bad (unrecognized) SMTP commands is exceeded
disconnectcommandlimit	(New in MS 7.0u1) Force SMTP session disconnect after a specified number of SMTP commands is exceeded
disconnectrecipientlimit	(New in MS 6.2) Force SMTP session disconnect after a specified number of recipients is exceeded
disconnectrejectlimit	(New in MS 6.2) Force SMTP session disconnect after a specified number of bad recipients (rejected recipients) is exceeded
disconnecttransactionlimit	(New in JES MS 6.2) Force SMTP session disconnect after a specified transaction limit is exceeded
domainetrn	Honor only those SMTP client ETRN commands that specify a domain
domainvrfy	Issue SMTP VRFY commands using full address
ehlo	Use EHLO on all initial SMTP connections
eightbit	Channel supports eight bit characters
eightnegotiate	Channel should negotiate use of eight bit transmission if possible
eightstrict	Channel should reject messages that contain unnegotiated eight bit data
expnallow	(New in JES MS 6.1) Explicitly enable support of the SMTP command EXPN
expndefault	(New in JES MS 6.1) Default handling of the SMTP command EXPN
expndisable	(New in JES MS 6.1) Disable support of the SMTP command EXPN
flagtransfer	Support private XDFLG SMTP extension; pass along delivery flags, such as trusting a subaddress
futurerelease	(New in MS 7.0) Enable source channel support for the future release SMTP extension
lmtp	Channel uses LMTP
lmtp_cr	Accept CR as an LMTP line terminator
lmtp_crlf	Require CRLF as the LMTP line terminator
lmtp_crorlf	Allow any of CR, LF, or CRLF as the LMTP line terminator
lmtp_lf	Accept LF as an LMTP line terminator
localvrfy	Issue SMTP VRFY command using local address
loopcheck	Enable support for the XLOOP SMTP extension (used to detect a type of message loop)
mailfromdnsverify	Verify that the domain specified on MAIL FROM: line is in the DNS
mtprioritiesallowed	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; other values adjusted to be within range
mtprioritiesrequired	(New in MS 8.0) Range of SMTP MT-PRIORITY values accepted during enqueue; messages with other values will be rejected.
nobinaryclient	(New in MS 6.3.) Disable BINARYMIME support in the SMTP client
nobinaryserver	(New in MS 6.3.) Disable BINARYMIME support in the SMTP server
nochunkingclient	(New in MS 6.3.) Disable CHUNKING support in the SMTP client
nochunkingserver	(New in MS 6.3.) Disable CHUNKING support in the SMTP server
noehlo	Never use the SMTP EHLO command

Functional group list of channel options

noflagtransfer	Disable XDFLG SMTP extension; do not pass along delivery flags
noloopcheck	Disable support for the XLOOP SMTP extension (used to detect a type of message loop)
nomailfromdnsverify	Do not perform DNS domain verification on the MAIL FROM: address
nosendetrn	Do not send SMTP ETRN command
nosmtp	Channel does not use SMTP
notary	Normal NOTARY support
noturn	Disable the SMTP TURN command
novrfy	Do not issue SMTP VRFY commands
noxclient	(New in MS 8.0) XCLIENT SMTP extension is disabled
passthrough	Do no message processing
recipientcutoff	Set a limit for the number of recipients-per-message-copy accepted on a channel; attempts to submit a message with more than this number of recipients will cause <i>all</i> recipients to be rejected
recipientlimit	Set a limit for the number of recipients-per-message-copy accepted on a channel; additional recipients will get a temporary rejection error
refuseehlo	RESTRICTED: Disable EHLO support in SMTP server
refusenotary	RESTRICTED: Disable DSN support in SMTP server
rejectsmtpplonglines	Reject incoming SMTP messages with illegally long lines
relay	Mark the channel as a relay channel
sendetrn	Send SMTP ETRN command
sevenbit	Channel does not support eight bit characters; eight bit characters must be encoded
silentetrn	Honor SMTP client ETRN commands, without echoing channel information
smtp	Channel uses SMTP
smtp_cr	Accept CR as an SMTP line terminator
smtp_crlf	Require CRLF as the SMTP line terminator
smtp_crorlf	Allow any of CR, LF, or CRLF as the SMTP line terminator
smtp_lf	Accept LF as an SMTP line terminator
sourcenosolicit	(New in MS 6.2) List of solicitation types not accepted
streaming	Specify degree of protocol streaming for channel to use
submit	Mark the channel as a submit-only channel
transactionlimit	(New in JES MS 6.1) Limit the number of transactions per SMTP session (messages per SMTP connection) accepted
truncatesmtpplonglines	Truncate incoming SMTP messages with illegally long lines
turn	RESTRICTED: Enable the SMTP TURN command
turn_in	RESTRICTED: Enable the SMTP TURN command for incoming connections; that is, the SMTP server will accept SMTP TURN commands
turn_out	RESTRICTED
vrfyallow	Provide informative responses to SMTP VRFY command
vrfydefault	Default responses to SMTP VRFY command, according to channel's HIDE_VERIFY option setting
vrfyhide	Provide obfuscatory responses to SMTP VRFY command
wrapsmtpplonglines	Wrap incoming SMTP messages with illegally long lines
xclient	(New in MS 8.0) XCLIENT SMTP extension is enabled, only one group of XCLIENT commands permitted
xclientrepeat	(New in MS 8.0) XCLIENT SMTP extension is enabled, and groups of XCLIENT commands are permitted
xclientsasl	(New in MS 8.0) XCLIENT SMTP extension is enabled, and LOGIN attribute is allowed
xclientsaslrepeat	(New in MS 8.0) XCLIENT SMTP extension is enabled, LOGIN attribute is permitted, and groups of XCLIENT commands are permitted
TCP/IP connections, DNS lookups, and SOCKS connections	
affinitylist	(New in MS 8.0) Enable affinity lookups, disable MX lookups
cacheeverything	Cache all connection information
cachefailures	Cache only connection failure information
cachesuccesses	Cache only connection success information
connectaliases	Do not rewrite addresses upon message dequeue
connectcanonical	Rewrite addresses upon message dequeue
daemon	Specify name of a gateway daemon to route to
defaultmx	Channel determines whether or not to do MX lookups from network
defaultnameservers	Consult TCP/IP stack's choice of nameservers
forwardcheckdelete	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, delete the name and use the IP address

Functional group list of channel options

forwardchecknone	Do not perform a forward lookup after a DNS reverse lookup
forwardchecktag	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, tag the name with *
identnone	Do not perform IDENT lookups; do perform IP to hostname translation; include both hostname and IP address in Received: header
identnonelimited	Do not perform IDENT lookups; do perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
identnonenumeric	Do not perform IDENT lookups or IP to hostname translation
identnonesymbolic	Do not perform IDENT lookups; do perform IP to hostname translation; include only the hostname in Received: header
identtcp	Perform IDENT lookups on incoming SMTP connections and IP to hostname translation; include both hostname and IP address in Received: header
identtcplimited	Do perform IDENT lookups; do perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
identtcpnumeric	Perform IDENT lookups on incoming SMTP connections, but do not perform IP to hostname translation
identtcpsymbolic	Perform IDENT lookups on incoming SMTP connections and IP to hostname translation; include only hostname in Received: header
interfaceaddress	Bind to the specified TCP/IP interface address
lastresort	Specify a last resort host
mailfromdnsverify	Verify that the domain specified on MAIL FROM: line is in the DNS
mx	TCP/IP network and software supports MX record lookups
nameservers	Consult specified nameservers rather than TCP/IP stack's choice (only for MX records as of MS 7.0)
nocache	Do not cache any connection information
nodns+	On OpenVMS, TCP/IP network does not support DNS (nameserver) lookups; on UNIX, merely disables MX lookups since on UNIX it is <code>nsswitch.conf</code> that controls consultation of nameservers
nomailfromdnsverify	Do not perform DNS domain verification on the MAIL FROM: address
nomx	TCP/IP network does not support MX lookups
nonrandommx	Do MX lookups; do not randomize returned entries with equal precedence
nosocks	Do not do SOCKS connections
port	Send to the specified TCP/IP port
randommx	Do MX lookups; randomize returned entries with equal precedence
spfhello	(New in JES MS 6.3) Perform an SPF lookup at the HELO/EHLO command
spfmailfrom	(New in JES MS 6.3) Perform an SPF lookup at the MAIL FROM command
spfnone	(New in JES MS 6.3) Do not perform any SPF lookups
spfrcptto	(New in JES MS 6.3) Perform an SPF lookup at the RCPT TO command
threaddepth	Number of messages triggering new thread with multithreaded SMTP client
TLS and SASL	
authpassword	(New in MS 7.0.5) Password for SMTP channel's client use of SMTP AUTH PLAIN
authrewrite	Use SMTP AUTH information in header
authusername	(New in MS 7.0.5) Username for SMTP channel's client use of SMTP AUTH PLAIN
disconnectbadauthlimit	(New in MS 6.2) Force SMTP session disconnect after a specified number of failed SMTP AUTH attempts
explicitssaslexternal	(New in MS 8.0) Disable automatic use of AUTH EXTERNAL at MAIL FROM
externalidentity	(New in MS 7.0.5) Identity for SMTP channel client use of SMTP AUTH EXTERNAL
implicitssaslexternal	(New in MS 8.0) Enable automatic use of AUTH EXTERNAL at MAIL FROM after successful negotiation of TLS
maysasl	Allow SMTP server and client SASL authentication
maysaslclient	(Effective as of MS 7.0-3.01) SMTP client attempts to use SASL authentication
maysaslserver	SMTP server offers SASL authentication
maytls	SMTP client and server allow TLS use
maytlsclient	SMTP client will attempt TLS use
maytlsserver	SMTP server allows TLS use
msexchange	Channel serves MS Exchange gateways
mustsasl	Must use SASL authentication
mustsaslclient	(Effective as of MS 7.0-3.01) SMTP client insists upon SASL authentication
mustsaslserver	SMTP server insists upon SASL authentication
musttls	SMTP client and server insist upon TLS use and will not transfer messages with remote sides that do not support TLS
musttlsclient	SMTP client insists upon TLS use and will not send messages to any remote SMTP server that does not support TLS use
musttlsserver	SMTP server insists upon TLS use and will not accept messages from any remote SMTP client that does not support TLS use
nomsexchange	Channel does not serve MS Exchange gateways

nosasl	SASL authentication not attempted or permitted
nosaslclient	SMTP client does not attempt SASL authentication
nosaslpassth	Do not pass along a MAIL FROM AUTH parameter
nosaslserver	SMTP server does not permit SASL authentication
nosaslswitchchannel	Do not switch channel upon successful SASL authentication
nosasltrustauth	(New in MS 7.0u3) Do not promote a MAIL FROM AUTH parameter to the MTA's authenticated originator address envelope field
notls	SMTP client and server neither attempt nor allow TLS use
notlsclient	SMTP client does not attempt TLS use when sending messages
notlsserver	SMTP server does not offer or allow TLS use when receiving messages
saslpassth	Pass along a MAIL FROM AUTH parameter
saslruleset	Specify the security configuration rule set to use for SASL transactions
saslswitchchannel	Switch to another channel when SASL authentication is successful
sasltrustauth	(New in MS 7.0u3) Promote any MAIL FROM AUTH parameter to the MTA's authenticated originator address envelope field
tlsswitchchannel	Switch to specified channel upon successful TLS negotiation

+Supported only on OpenVMS.

33.3.3 Addresses channel options

There are many channel options relating to address handling. These are some of those most directly relating to address handling. See also the [Long address lists or headers channel options](#) for special handling of large numbers (long lists) of addresses, and the [Headers channel options](#) as a number of channel options relate to insertion or propagation of addresses in header lines.

33.3.3.1 _733 Option

The legacy configuration 733 channel option has been replaced in Unified Configuration by [percents](#).

33.3.3.2 _822 Option

The legacy configuration 822 channel option has been replaced in Unified Configuration by [sourceroute](#).

33.3.3.3 Envelope recipient validity checks ([acceptalladdresses](#), [acceptvalidaddresses](#))

When specified on a source channel, the [acceptalladdresses](#) option causes all envelope recipient addresses to be accepted unconditionally. Rather than returning errors during the SMTP session, a [delivery status notification](#), or DSN, will be returned later if the message cannot be delivered to the specified recipient.

In contrast, setting the [acceptvalidaddresses](#) option on a source channel causes all envelope recipient addresses to be validated prior to being accepted by the MTA. If an address fails to validate an appropriate error will be returned to the client. [acceptvalidaddresses](#) is the default.

The [acceptalladdresses](#) channel option is useful when dealing with clients that fail to report SMTP errors correctly to the user. In such cases a DSN, while providing less immediate indication that there's a problem, may be the only way to convey accurate error information to the user.

Unconditional address acceptance should be used with extreme care; in particular, it should never be used on a source channel that accepts unauthenticated transactions from the Internet. The problem with such usage is that spammers will attempt to send mail to invalid addresses, which will be accepted and later result in a DSN being returned. Since envelope From addresses on spam are commonly forged, this will turn the system into a major source of blowback spam and is likely to result in blacklisting and other operational issues.

These options were first added in JES MS 6.1.

33.3.3.4 Filling in missing header addresses (`addlineaddr`, `noaddlineaddr`)

Situations can arise where messages are submitted to the MTA without proper originator and recipient address information in the message header. In such cases there may be additional information available in a nonstandard form, either in the header or elsewhere, that can be used to reconstruct the missing header information. When the `addlineaddr` option is placed on a source channel it instructs submission agents to use whatever additional information is available to reconstruct the missing information. `noaddlineaddr` is the default and prevents this from happening.

Currently the only submission agent that supports `addlineaddr` is the VMS MAIL interface in PMDF. This option has no effect on any Messaging Server channel.

33.3.3.5 Controlling Sender Rewriting Scheme (SRS) rewriting (`addresssrs`, `noaddresssrs`, `destination`, `nodestination`, `sourcesrs`, `nosourcesrs`)

(New in JES MS 6.3p1.) Sender Rewriting Scheme (SRS) rewriting is used by sites providing autoforwarding services to encapsulate MAIL FROM addresses so as to avoid running afoul of [Sender Permitted From \(SPF\)](#) checks. In order for an address to undergo SRS rewriting, SRS support must be configured and the following three conditions must be met: (1) The current source channel must be marked with the `sourcesrs` option, (2) The current destination channel must be marked with the `destination` option, and (3) Rewriting the MAIL FROM address must have matched a channel marked with the `addresssrs` option. `noaddresssrs`, `nodestination` and `nosourcesrs` are the default for all channels.

33.3.3.6 Address type flags (`addrtypescan`, `addrtypescanbccdefault`, `noaddrtypescan`)

(New in 7.0.5.) If the `addrtypescan` channel option is set, then RCPT TO addresses (that is, envelope recipients) are compared with header recipient fields (To:, Cc:, Bcc:, Resent-To:, Resent-Cc:, and Resent-Bcc:). When a match is found, that fact is recorded in the delivery flags associated with that envelope recipient. Those flags are then used when generating the [report part](#) of Microsoft[®] Exchange 2007 [envelope journaling](#) archive messages, distinguishing between various types of envelope recipient addresses.

`addrtypescanbccdefault` operates in the same way as `addrtypescan`, except that when no matches are found for a given address, that address is assumed to be a blind carbon (Bcc:) recipient. This option should only be used when it is certain the messages have come directly from a client that implements Bcc: by simply omitting the blind carbon recipient from the header and which doesn't support any form of local mailing lists. Use in any other context is *guaranteed* to result in incorrect types being attached!

`noaddrtypescan` is the default.

Also note that since the MTA's delivery flags are used to store this information, the MTA's delivery flag transfer facilities may be used to transport this information between MTAs; see the [deliveryflags channel option](#).

33.3.3.7 Force "detour" routing of hosted users (`aliasdetourhost`, `aliasoptindetourhost`)

The (new in iMS 5.2p2, and JES MS 6.1) `aliasdetourhost` channel option allows source-channel-specific overriding of hosted users' `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. It allows better configuration and use of "intermediate filtering" sorts of channels and third party filtering hosts.

The `aliasdetourhost` channel option takes a single host/domain name as an argument. When specified on a source channel, this channel option causes alias expansion of addresses stored in LDAP to stop (short-circuit) just prior to the point where `mailHost` (more precisely, the attribute named by the `ldap_mailhost` MTA option) information is checked. The host specified by the `aliasdetourhost` channel option is used as the (assumed to be non-local) `mailHost`. That is, a source route containing the specified host is added to the address (just as if a non-local `mailHost` had been found) and processing continues onward from that point. Note that in particular, this forced use of the `aliasdetourhost` specified host as a non-local `mailHost` stops further expansion of the alias for purposes of things such as application of user forwarding and Sieve filter application (which normally would occur subsequently during alias expansion when a user's real `mailHost` is this MTA).

Thus use of `aliasdetourhost` on an incoming channel lets the MTA do address validation (check that an incoming address corresponds to a valid user entry), while "delaying" complete expansion and processing (in particular, forwarding and Sieve evaluation) of the valid local recipient addresses. This combination of effects is potentially very useful.

A typical application of this channel option is for purposes of "detouring" messages through a special channel or host, most often for purposes of spam/virus filtering. It is often used in conjunction with use of an "alternate" conversion channel for such "detour" purposes, where the "alternate" conversion channel approach is used to handle cases of non-local recipient addresses, while `aliasdetourhost` is used to handle cases of local-to-this-`mailHost` recipient addresses. (Use of an "alternate" conversion channel approach for a routing "detour" on local-to-this-`mailHost` recipient addresses incurs various problems, in particular in the areas of forwarding and Sieve filter evaluation timing. It is desirable to delay Sieve filter evaluation until after the "detour" - for instance, so that Sieve filters can look for headers added by the "detour" host. It is also desirable to delay application of user forwarding until after the "detour", to avoid potential duplication of the forwarding. Such a delay in the final parts of user alias expansion is exactly what `aliasdetourhost` can be used to achieve.)

The (new in JES MS 6.2p4) `aliasoptindetourhost` option has the same function as `aliasdetourhost`, except that it only applies for users in LDAP who have "opted-in" via whatever user attribute is named by the `ldap_detourhost_optin` MTA option, or whatever domain attribute is named by the `ldap_domain_attr_detourhostoptin` MTA option. The argument of the `aliasoptindetourhost` channel option specifies a list of detour hosts separated by commas. The value(s) of the `optin` attribute are compared with the list; the first match will be used as the "override" `mailHost` for any users who are "opted-in". However, any attribute that doesn't contain at least one period (which would be necessary to match a

legitimate mail host) is treated as an effective wildcard; the first host from the list will be used in this case.

Finally, if the option value matches the special value specified by the `aliasdetourhost_null_optin` MTA option it will simply be ignored. This mechanism is provided to accommodate provisioning systems that insist on every known attribute having a value. Omitting the attribute value entirely is the preferred method for disabling detour processing, however.

33.3.3.8 Local address processing control (`aliaslocal`)

Normally only addresses rewritten to the local channel undergo `alias` processing. The `aliaslocal` channel option may be associated with a channel to cause addresses rewritten to that channel to undergo alias expansion.

33.3.3.9 Sources of alias information (`aliasmagic`)

`Alias` expansion is performed by consulting various sources of alias information one after another until a match is found. The `aliasmagic` channel option provides a means to control what sources are consulted and in what order. The argument to `aliasmagic` is an unsigned decimal value. Each decimal digit value specifies a different source of alias information. Possible values are:

Table 33.5 Alias_magic digits

Value	Meaning
0	No operation
1	Personal alias database
2	Local name table aliases
3	System <code>alias database</code>
4	System <code>alias file</code>
5	Special aliases
6	<code>alias_url0</code>
7	<code>alias_url1</code>
8	<code>alias_url2</code>
9	<code>alias_url3</code>

The digits of `aliasmagic` are processed from left to right. Processing stops when a match occurs. The default value for `aliasmagic` is given by the `alias_magic` MTA option.

33.3.3.10 Wildcard alias lookups (`aliaswild`)

Setting the `aliaswild` option on a destination channel causes an address of the form `*@domain` to be checked in the `alias file` and `alias database` if the regular lookup for `localpart@domain` fails. This lookup is performed before checking in the alias file and alias database for local parts without domains.

The effect of this setting is the same as if bit 2 (value 4) of the `alias_domains` MTA option is set; the only difference is that `aliaswild` only applies to a specific destination channel.

33.3.3.11 Authenticated originator information processing (authrewrite)

The `authrewrite` option may be used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers. Normally the SMTP AUTH information is used (specifically, the user's canonical e-mail address, that is, the value of the `mail` attribute, found when looking up the user for authentication), though this may be overridden via the [FROM_ACCESS mapping](#). `authrewrite` takes a required bit-encoded integer value as an argument, according to the following table:

Table 33.6 `authrewrite` option values

Bit	Value	Usage
0-3	1	Add a Sender: header line, or a Resent-sender: header line if a Resent-from: or Resent-sender: was already present, containing the AUTH originator
0-3	2	Add a Sender: header line containing the AUTH originator
0-3	3	Use the AUTH_REWRITE mapping table , probing with any Resent-Sender: and Resent-From: info if present, and otherwise probing with Sender: and From: info
0-3	4	Use the AUTH_REWRITE mapping table , probing with Sender: and From: info
0-3	5	Add a From: header line, or a Resent-From: header line if a Resent-From: or Resent-Sender: was already present, containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!
0-3	6	Add a From: header line containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!
4	16	(New in 6.2) If set, apply the AUTH_REWRITE mapping table , even if SMTP AUTH has not been used
5	32	(New in 6.2) If set, probes to AUTH_REWRITE include the source-channel as a prefix field, separated by a vertical bar character from the rest of the probe string; that is, when this bit is set then probes take the form: <i>source-channel env-from [resent-]sender [resent-]from auth-originator</i>
6	64	(New in 7.2-7.02.) If set, use the rewritten version of the envelope from address in constructing the AUTH_REWRITE probe.
7	128	(New in 7.2-7.02.) If set, use the canonical version of the envelope from address in constructing the AUTH_REWRITE probe. Bit 6 (value 64) is a no-op if this bit is set.
8	256	(New in 7.3-11.01.) If set, add the value of the AUTH parameter from the SMTP MAIL FROM command to the AUTH_REWRITE probe, appearing just after the authorized originator address.

33.3.3.11.1 AUTH_REWRITE mapping table

Certain values of the [authrewrite channel option](#) cause the `AUTH_REWRITE` mapping table to be consulted to allow for more complex decision making and alterations of addresses. And bits of `authrewrite` also affect the form of probe to the `AUTH_REWRITE` mapping table.

Probes for the `AUTH_REWRITE` mapping table normally have the following format:

```
source-channel|env-from|[resent-]sender|[resent-]from|auth-originator|auth-parameter
```

Note that the `source-channel` field and its vertical bar suffix is only present if (new in JES MS 6.2) bit 5 (value 16) is set in the `authrewrite` argument, and `auth-parameter` and its vertical bar prefix are only present if (new in 7.3-11.01) bit 8 (value 256) is set in the `authrewrite` argument.

With `authrewrite 3`, the probes preferentially use any Resent-Sender: or Resent-From: header line values present, whereas with `authrewrite 4` the probes always use Sender: and From:. (Note that normally the `AUTH_REWRITE` mapping table is only consulted when

a submission has included SMTP AUTH info; that is, in order for the AUTH_REWRITE mapping table to be consulted not only must the relevant incoming channel be marked with an `authrewrite` value of 3 or 4, but also the submission included use of the SMTP AUTH command. However, if bit 4 (value 16) is set in the `authrewrite` channel option's argument, then AUTH_REWRITE will be consulted even for non-authenticated submissions.)

New in 7.2-7.02, bit 6 (value 64) of `authrewrite` will, if set, cause a rewritten version of the envelope from address to be used for the `env-from` address in the probe as opposed to the original form given in the SMTP MAIL FROM command. The specific rewritten form used is controlled by bit 7 (value 128): If set the canonical form return address will be used, if clear the normally rewritten form will be used instead. These rewritten forms are useful when accessing checking is done using the AUTH_REWRITE mapping in order to prevent envelope from forgery by authenticated users.

New in 7.0.5, if bit 9 (value 512) of `authrewrite` is set, the final tag set by the `*_ACCESS` mappings will be prefixed to the AUTH_REWRITE mapping probe.

As of the 8.0 release, the following input flags will be set:

- \$A if SASL authentication has succeeded
- \$E if EHLO (EMSMTP) was used
- \$L if LHLO (LMTP) was used
- \$P if POP-before-SMTP was used
- \$R if this is an internal channel enqueue operation, *i.e.*, from a [conversion](#), [process](#), [reprocess](#), or similar sort of channel
- \$T if a SSL/TLS security layer has been negotiated

If the mapping table output contains a \$J, \$j, \$K, or \$k, then the envelope From address is replaced with the specified string. If the mapping table output contains a \$Y, \$y, \$T, or \$t, then a Sender: header line is added (if `authrewrite 3` was specified and if a Resent-Sender: or Resent-From: was already present, then a Resent-Sender: header line is added instead of a Sender: header line) containing the specified string.

If the mapping table output contains a \$Z or \$z, then a From: header line is added (a Resent-From: in the case of `authrewrite 3` and a Resent-From: or Resent-Sender: header line already being present) containing the specified string. (Such replacing of the From: header address is **NOT RECOMMENDED** and **CONTRARY TO INTERNET STANDARDS** and quite likely to **HARM** the overall security of your users. It should almost **NEVER** be done: **THIS MEANS YOU!** Despite the wishes and mistaken notions of many sites and users, the From: header line, in Internet e-mail, is **NOT INTENDED** to represent the "real" originator of a message; it is intentionally defined permitting alternate usages.)

New in 7.3-11.01, if a \$O is specified, then another vertical-bar-separated string will be read from the mapping result string and used to set or override the value of the SMTP AUTH parameter for the current transaction. The `saslpassauth` channel option may then be applied to the destination channel to cause this value to be propagated as an AUTH parameter on the SMTP MAIL FROM command.

New in JES MS 6.2, if a \$N is specified, then the message will be rejected. Optional rejection text may be specified after another vertical bar character, |. And as of JES MS 6.3, \$X may also

be used to specify the extended error code (specified before the \$N text, separated by a |) in the form x.y.z. In the absence of such optional text and optional extended error code, the default text "invalid originator address used" and default extended error code 5.7.0 will be used.

When using multiple such flags, separate the string arguments with the vertical bar character, |, and specify the string arguments in the order listed in the paragraph above; that is,

```
$J$Y$Z#env-from|sender|from-header
```

or

```
$X$N|error-code|rejection-text-string
```

Technically, one could use all five flags in the same entry, though it does not seem likely to be useful:

```
$J$Y$Z$X$N#env-from|sender|from-header|error-code|rejection-text-string
```

33.3.3.12 Interpretation of local parts (bangoverpercent, nobangoverpercent, bangonly, percentonly, nobangorpercent)

The local parts of addresses are interpreted in accordance with the rules specified in [RFC 5322](#) and [RFC 976](#). However, there may be occasions when interpretation of embedded routing information is inappropriate. Additionally, ambiguities in the treatment of certain composite addresses that are not addressed by these standards. In particular, the interpretation of exclamation points and percent signs appearing in the local parts of addresses in domains the MTA has administrative authority over are not specified in any standard. These characters are sometimes used to provide a form of explicit multihop routing comparable to source routes.

Several source channel options are provided to control the interpretation of these characters. `nobangorpercent` disables special interpretation of both characters completely. `bangonly` treats local parts of the form A!B as a route from routing host A to user B; percent signs have no special meaning. `percentonly` treats local parts of the form A%B as a route from routing host B to user A; exclamation points have no special meaning.

If both characters are used to specify routing a question arises as to which one has precedence. 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 it is all right for mailers to interpret addresses using the latter set of conventions, it does not say that such an interpretation is required. In fact, some situations may be better served by the former interpretation.

In any case, the `bangoverpercent` channel option forces the former A!(B%C) interpretation. `nobangoverpercent` forces the latter (A!B)%C interpretation. `nobangoverpercent` is the default for all channels.

33.3.3.13 Address types and conventions (sourceroute, percents, bangstyle, header_822, header_733, header_uucp)

This group of destination channel options controls what types of addresses will be used in messages queued to the channel. A distinction is made between the addresses used in the transport layer (the message envelope) and those used in message headers.

The `sourceroute` option is the default and specifies that source routed envelope addresses should be used. This channel supports full [RFC 5322](#) format envelope addressing conventions including source routes. The keyword `822` was a supported synonym for `sourceroute` in old configuration files but is not supported in XML-based configurations.

`percents` specifies that percent sign addressing conventions should be used in the envelope. The channel supports full [RFC 5322](#) format envelope addressing with the exception of source routes; source routes will be rewritten using percent sign conventions instead. The channel keyword `733` was a supported synonym for `percents` in old configuration files but is not supported in XML-based configurations.

Use of percent sign addresses on an SMTP channel will result in these conventions being carried over to the transport layer addresses in the SMTP envelope. This may violate [RFC 5321](#). Only use percent sign addressing conventions when you are sure they are necessary.

The `bangstyle` channel option specifies that this channel uses addresses that conform to [RFC 976](#) bang-style address conventions in the envelope (*i.e.*, this is a UUCP channel). The keyword `uucp` was a supported synonym for `bangstyle` in legacy configuration files but is not supported in unified configurations.

The `header_822` channel option specifies that source routes should be used in header addresses. This channel supports full [RFC 5322](#) format header addressing conventions including source routes. This is the default if no other header address type option is specified.

The `header_733` channel option specifies that percent sign addresses should be used in the header. This channel supports [RFC 5322](#) format header addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead.

Note that the use of 733 address conventions in message headers may violate [RFC 5322](#) and [RFC 976](#). Only use this option if messages are known to contain source route addresses in the header and you are sure that the channel connects to a system that simply cannot deal with source route addresses.

`header_uucp` specifies that bang-style addresses should be used in the header. The use of this option is *not* recommended. Such usage grossly violates [RFC 976](#).

33.3.3.14 Recipient validity date check (`checkrrvs`, `ignorerrvs`)

The `checkrrvs` and `ignorerrvs` source channel options, for support of [RFC 7293](#), are new in 8.0. `ignorerrvs` is the default.

`ignorerrvs` means that the SMTP server will not offer or support the RRVVS SMTP extension. In particular, client attempts to use an RRVVS parameter in a RCPT TO command will cause an error, and the MTA will ignore any Require-Recipient-Valid-Since: header line.

`checkrrvs` when set on an SMTP source channel means that the SMTP server offers and supports use of the SMTP RRVVS extension. In particular, the MTA will check for a valid date for recipient mailbox ownership, whether specified (preferentially) in a RRVVS parameter in the RCPT TO command, or in a Require-Recipient-Valid-Since: header field, and check for a valid date for the domain in the recipient address.

If the `checkrrvs` check fails on the recipient mailbox address, the recipient will be rejected with the SMTP error:

```
550 5.7.15 account information on file is older than actual user account
```

or alternate text as controlled by the `error_text_wrong_account` MTA option; if the `checkrrvs` check fails due to the domain creation date, the recipient will be rejected with the SMTP error:

```
550 5.7.18 domain owner has changed
```

or alternate text as controlled by the `error_text_wrong_domain` MTA option.

33.3.3.15 Clone messages to alternate destination (`clonehosts`)

A commonly requested capability is to "clone" all messages that meet some criteria and send them to an alternate destination. This can be accomplished in a variety of ways, including the Sieve "capture" action, the `MESSAGE-SAVE-COPY` mapping, the `FORWARD` mapping, and various sorts of address rewriting tricks, and all of these mechanisms have different characteristics as well as different advantages and disadvantages.

In the specific case when the desire is to clone all messages sent to a particular channel to another channel while preserving the initial address that expanded to that channel, none of the previously mentioned methods provide that specific result. (A `capture` action in a `destination channel Sieve script` can capture the message, but not the initial address. `MESSAGE-SAVE-COPY` has similar limitations and also requires playing queue management games.)

The `clonehosts` channel option provides this result. It accepts a single argument: A space-separated list of host names. When given a `clonehosts` setting of "host1 host2 host3" and a message sent to the addresses `initial1` and `initial2`, both of which expanded to one or more final recipient addresses destined to that channel, this setting will add the recipient addresses:

```
@host1:initial1
@host2:initial1
@host1:initial2
@host2:initial2
```

33.3.3.16 Host name to use when correcting incomplete addresses (`remotehost`, `noremotehost`, `defaulthost`, `nodefaulthost`)

The MTA often receives from misconfigured or in-compliant mailers and SMTP clients addresses which do not contain a domain name. The MTA, showing at least some respect for standards, must attempt to make such addresses legal before passing the message along. The MTA does this by appending a domain name to the address (*e.g.*, appends "@acme.com" to "mrochek"). For envelope To addresses missing a domain name, the MTA normally assumes that the local host name should be appended -- however, see the detailed discussion of the `defaulthost` option below for some exception cases. (Exceptions: unless the `defaulthost` option is used and either the channel is a `submit` channel so that the `defaulthost` option's

first argument applies also to envelope To addresses, or a second parameter (that has at least one period in it) is specified for the `defaultHost` option.) However for other addresses, such as From: addresses, in the case of the MTA's SMTP server there are at least two reasonable choices for the domain name: the local MTA host name, or 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, for email from arbitrary remote sources, 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 mail client such as a POP or IMAP client that uses SMTP to post messages. Or if lightweight mail clients such as POP or IMAP clients "ought" to have their own specific domain name which is not that of the local host, then adding 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 `remoteHost` channel option specifies that the remote host's name should be used. The `noRemoteHost` channel option specifies that the local host's name should be used and that any existing `remoteHost` setting be reset. The default to neither use the remote host's name nor reset any existing setting.

The `defaultHost` channel option is used to specify a particular host name to append to incoming bare usernames; it must be followed by the domain name to use in completing addresses (in envelope From: and in headers) coming in that channel. An optional second domain name (that contains at least one period) may be specified to use in completing envelope To addresses. `noDefaultHost` is the default. `defaultHost` overrides `remoteHost` if both are set.

Note that the `switchChannel`, `saslSwitchChannel`, `tlsswitchChannel` and various other options can be used to associated incoming SMTP connections with a particular channel other than the one the SMTP server uses by default. The default and remote host settings on the final channel that's switched to will be the ones that are used. Note, however, that once `remoteHost` has been activated only an explicit `noRemoteHost` setting will deactivate it. This facility can be used to group remote mail clients on a channel where they will receive proper treatment. Alternatively, it is an unconditionally simpler proposition to deploy standards-compliant remote mail clients (even if a multitude of in-compliant clients are in use) rather than attempting to fix the network-wide problem on your MTA hosts.

33.3.3.17 `dequeue_removeRoute` Option

The legacy configuration `dequeue_removeRoute` channel option has been replaced in Unified Configuration by `dequeueRemoveRoute`.

33.3.3.18 Removing source routes (`dequeueRemoveRoute`, `enqueueRemoveRoute`)

The `dequeueRemoveRoute` channel option, when placed on a destination channel, causes source routes to be stripped from envelope recipient addresses when the channel dequeues messages (but after the channel has determined how to route the message). For instance, on a `dequeueRemoveRoute TCP/IP channel` that is not a `daemon` channel, the source route host is used to determine to what remote host to connect, but is stripped from the envelope To addresses before they are presented to that remote host. In particular, this channel option may be useful at sites that use the `mailHost` LDAP attribute (more precisely, whatever LDAP attribute is named by the `ldap_mailHost` MTA option) to direct messages (via source routes,

`@mailhost:orig-address`), to NMS systems or other systems that do not support source routes; the `dequeue_remove_route` channel option would be placed on a special TCP/IP channel set up to send to such an NMS system. But this channel option should not be used on a general channel where source routing in addresses may need to be preserved.

The `enqueue_remove_route` option, when placed on a destination channel, causes source routes to be stripped from recipient addresses enqueued to that channel *after* the regular address rewriting has been performed. Thus the `enqueue_remove_route` channel option may be useful in cases where a `mailHost` or `mailRoutingSmartHost` LDAP attribute (more precisely, whatever LDAP attribute is named by the `ldap_mailhost` or `ldap_domain_attr_smarthost` MTA option) has been set for a user or domain merely in order to force a particular channel match to some special channel during rewriting, but where the host specified in such an attribute is not relevant for actual mail delivery.

The obsolete `dequeue_remove_route` and `enqueue_remove_route` options are aliases for `dequeue_remove_route` and `enqueue_remove_route`, respectively. These obsolete options are not supported in unified configuration mode.

33.3.3.19 `enqueue_remove_route` Option

The legacy configuration `enqueue_remove_route` channel option has been replaced in Unified Configuration by `enqueue_remove_route`.

33.3.3.20 Routing information in addresses (`exproute`, `noexproute`, `improute`, `noimproute`)

The ideal 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 attainable in many cases. The usual exception occurs when a channel connects to one or more systems that are not known to the rest of the world (*e.g.*, internal machines on a private network). Addresses for systems on this channel may not be legal on remote systems outside of the site. If such addresses are to be made reliable, 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` channel option (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 option is specified on a channel, the MTA will add 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 MTA option `exproute_forward` can be used to restrict the action of `exproute` to backward-pointing addresses if desired.

Another scenario occurs when the MTA connects to a system via a channel that cannot perform proper routing for itself. In this case all addresses associated with other channels need to have routing inserted into them when they are used in mail sent to the channel that connects to the incapable system.

Implicit routing and the `improute` channel option are 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`. `noimproute`, the default, specifies that no routing information should be added to addresses in messages going out on the specified channel.

The `improute_forward` MTA option can be used to restrict the action of `improute` to backward-pointing addresses if desired.

The `exproute` and `improute` channel options should be used sparingly. It makes 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.

33.3.3.21 Local-channel-like behavior (`localbehavior`, `nolocalhostbehavior`)

Use of `localbehavior` or `nolocalhostbehavior` is RESTRICTED: do not use unless explicitly instructed to do so by Oracle.

The `localbehavior` channel option enables certain local-channel-like behaviors, including subsuming `aliaslocal` and `routelocal` effects. Explicit setting of `localbehavior` or `nolocalhostbehavior` is not usually appropriate; instead, channel defaults, optionally modified in specific ways via options such as `aliaslocal` or `routelocal`, should be used.

33.3.3.22 Legalizing messages that lack any recipient headers (`missingrecipientpolicy`)

[RFC 822](#) (Internet) messages are required to contain a recipient header line: a `To:`, `Cc:`, or `Bcc:` header line. A message without any such header line is illegal according to [RFC 822](#). Nevertheless, some broken user agents and mailers (*e.g.*, many older versions of `sendmail`) will emit such illegal (per [RFC 822](#)) messages.

Note that [RFC 2822](#), the update to [RFC 822](#), relaxes the [RFC 822](#) requirement and allows submitted messages to be lacking in any recipient header line. However, unless it is certain that *all* the MTAs and MUAs that may ever handle a message in fact conform to [RFC 2822](#) (rather than the older [RFC 822](#)), it is unwise to emit a message lacking all recipient header lines, since the behavior of an [RFC 822](#)-compliant MTA or mail user agent may be undesirable when encountering a message that is, from its point of view, illegal--results may include rejection of such a message, potentially undesired exposure of recipient information such as recipients intended as `Bcc:` recipients, *etc.*

The `missingrecipientpolicy` channel option takes an integer value specifying what approach to use for such messages; the default value, if the channel option is not explicitly present, is to use the MTA option `missing_recipient_policy` value (which itself defaults to 0, if not set, which as of JES MS 6.2 is equivalent to a value of 1 meaning that messages are passed through unchanged--in JES MS 6.0 and JES MS 6.1 the default value of 0 had been equivalent to a value of 2 meaning that envelope `To:` addresses are placed in a `To:` header).

Table 33.7 `missingrecipientpolicy` MTA option values

Value	Action
-------	--------

0	Use current best practices to resolve the situation. Prior to 6.2 this was the same as 2, in 6.2 and later it is the same as 1.
1	Pass the illegal-per-RFC 822 (though legal per RFC 2822) message through unchanged.
2	Place envelope To: recipients in a To: header.
3	Place all envelope To: recipients in a single Bcc: header.
4	Generate an empty group construct (<i>i.e.</i> , ;) To: header line. The phrase used in the group construct is controlled by the <code>missing_recipient_group_text</code> MTA option, so for instance "To: Recipients not specified: ;".
5	Generate a blank Bcc: header.
6	Reject the message (with a "554 5.6.0 Error writing message - message is missing required recipient header fields" error).

Note that the `missing_recipient_policy` MTA option can be used to set an MTA system default for this sort of behavior.

33.3.3.23 Restricted mailbox encoding (`restricted`, `unrestricted`, `norestricted`)

Some mail systems have great 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:

```
"freed, ned"@ymir.claremont.edu
```

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 and 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:

```
freed#m#_ned@ymir.claremont.edu
```

The `restricted` channel option 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` channel option tells the MTA that this channel wants to see quoted addresses and that any restricted encodings should be decoded.

The `norestricted` channel option tells the MTA not to perform RFC 1137 encoding and decoding. `norestricted` is the default.

IMPORTANT NOTE: The `restricted` and `unrestricted` channel options should be applied to the channels that connects to systems unable to accept quoted or restricted local-parts respectively. They should *not* be applied to the channels that actually generate the quoted or restricted local-parts! (It is assumed that a channel capable of generating such an address is also capable of handling such an address.)

33.3.3.24 Channel-specific use of address reversal (`reverse`, `noreverse`)

The `reverse` channel option tells the MTA that addresses in messages enqueued to the channel should be checked against and possibly modified by [address reversal](#) (that is, modified by `reverse_url` controlled LDAP lookups, or the address reversal database, or the [REVERSE mapping](#)). `noreverse` exempts addresses in messages queued to the channel from address reversal processing. The `reverse` channel option is the default.

Note that, due to [intended critical side-effects of `reverse_url` LDAP lookups](#), side-effects that will not occur if the `noreverse` channel option is used, typical Messaging Server sites **should not** use the `noreverse` channel option.

33.3.3.25 Channel-specific rewrite rules (`rules`, `norules`)

The `rules` channel option tells the MTA to enforce channel-specific rewrite rule checks, and can be placed on source channels and on destination channels. When placed on a source channel, `rules` allows [source channel-specific effects from `\$N` or `\$M` control sequences in a rewrite rule template](#) to take effect when that source channel is enqueueing a message and rewriting addresses. When placed on a destination channel, `rules` allows [destination channel-specific effects from `\$C` or `\$Q` control sequences in a rewrite rule template](#) to take effect for addresses in messages being enqueued to that destination channel. `rules` is the default. `norules` tells the MTA not to check. These two channel options are usually used for debugging and are rarely used in actual applications.

33.3.3.26 Personal names in address message headers (`personalinc`, `personalmap`, `personalomit`, `personalstrip`, `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, `sourcepersonalstrip`)

The MTA only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process personal names (strings preceding angle-bracket-delimited addresses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

On destination channels, this behavior is controlled by the use of the `personalinc`, `personalmap`, `personalomit`, and `personalstrip` channel options. `personalinc` tells the MTA to retain personal names in the headers. It is the default. `personalmap` tells the MTA to apply the [PERSONAL_NAMES](#) mapping table to personal names appearing in addressing header lines if such a mapping table exists, while if no such mapping table exists then `personalmap` is equivalent to `personalstrip`. `personalomit` tells the MTA to remove all personal names. And finally, `personalstrip` tells the MTA to strip any nonatomic characters from all personal name fields.

On source channels, this behavior is controlled by the use of a `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, or `sourcepersonalstrip` channel option. `sourcepersonalinc` tells the MTA to retain personal names in the headers. It is the default. `sourcepersonalmap` tells the MTA to apply the [PERSONAL_NAMES](#) mapping table to personal names appearing in addressing header lines if such a mapping table exists, while if no such mapping table exists then `sourcepersonalmap` is equivalent to `sourcepersonalstrip`. `sourcepersonalomit` tells the MTA to remove all personal

names. And finally, `sourcepersonalstrip` tells the MTA to strip any nonatomic characters from all personal name fields.

These options can be applied to any channel.

33.3.3.27 Short circuiting rewriting of routing addresses (`routelocal`)

The `routelocal` channel option 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, or with an address embedded within quotes as the local-part of the address) will be simplified. (The "l" channel defaults to `routelocal` behavior.)

For instance, if `domain.com` is a domain that rewrites to a channel marked with the `routelocal` channel keyword, then any of the addresses

```
somewhere.else.com!user@domain.com
user%somewhere.else.com@domain.com
@domain.com:user@somewhere.else.com
"user@somewhere.else.com"@domain.com
```

will be rewritten to simply `user@somewhere.else.com`.

Use of this keyword on "internal" channels, such as internal [TCP/IP channels](#), can potentially allow simpler configuration of [SMTP relay blocking](#).

However, note that this keyword should not be used on channels that may require explicit `@mailhost` source routing or other routing, such as typical Oracle Messaging Server MTA internal TCP/IP channels.

33.3.3.28 Subaddresses and alias matching (`subaddressexact`, `subaddressrelaxed`, `subaddresswild`)

A *subaddress* consists of extra detail information in the [RFC 5322](#) "local-part" of an address (the portion to the left of the "@" sign); the subaddress is typically encoded into the local-part by using a [separator character such as the plus character, +](#), and is subject to site-specific interpretation. (See for instance the discussion in the introduction of [RFC 5233](#), [Sieve: Subaddress Extension](#).) Use of subaddresses can be a convenient way to, *e.g.*:

- Request delivery directly to a named folder.
- Indicate that a message is being received [due to membership of some mailing list](#).
- Request other special delivery handling, such as delivery to a voice mailbox.

In regard to subaddresses, the Messaging Server `ims-ms` and `tcp_lmtps*` 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 Messaging Server Message Store delivery code considers the portion of the mailbox after the plus character a *subaddress*; if either the subaddress is ["trusted"](#) (as in the case of a subaddress added due to a [Sieve filter "fileinto" action](#) and confirmed for the channel via the `fileinto` channel option), or the folder has the IMAP post ACL set, *then* such channels may treat a subaddress as a request to deliver directly into the correspondingly named folder.

Subaddresses also affect the lookup of aliases by the local channel and the lookup of aliases by any channel marked with the [aliaslocal](#) channel option, 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](#). As of JES MS 6.1, the subaddress support in aliases includes [alias_urlN](#) alias lookups; that is, as of JES MS 6.1, the `subaddress*` channel options apply for `alias_urlN` lookups.

The [subaddressexact](#) channel option 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 wildcarded comparisons or comparisons with the subaddress removed) will be performed. The [subaddresswild](#) channel option 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+*`. (For wildcarding the entire localpart, not just the subaddress, see the [alias_domains](#) MTA option.) The [subaddressrelaxed](#) channel option 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 form

```
name:    newname+*
```

will match either `name` or `name+subaddress`, transforming a plain name to `newname`, and transforming `name+subaddress` to `newname+subaddress`. The [LDAP entry equivalent](#) with `subaddressrelaxed` set, to get the "transfer" of the subaddress to the forwarded-to address, would be to set:

```
mailDeliveryOption: forward
mailForwardingAddress: newname+*@newdomain
```

The default is `subaddressrelaxed`.

Thus the `subaddresswild` channel option or the `subaddressrelaxed` channel option may be useful when [aliases](#) or a directory channel are in use yet users wish to receive mail addressed using arbitrary subaddresses. These channel options obviate the need for a separate entry for every single subaddress variant on an address.

For the Messaging Server MTA, these channel options make sense on the L channel as a destination (or rather, as an alias application) channel.

New in 7.0.5, a `subaddress*` channel option setting on a source channel will affect [address reversal](#) performed on messages coming in that source channel. Previously, the presence of a subaddress would prevent address reversal from occurring. (This long-standing behavior was a remnant of the past when if a user was sophisticated enough to put on a subaddress, one might presume that the user was sophisticated enough to have already specified the exact address that they wanted to send from -- so altering such an address wouldn't be necessary and indeed would be dubious. However, nowadays many other [behaviors and side-effects are triggered via address reversal](#) so matching regardless of subaddress is typically desirable; the old assumption that reversal was not desirable in such cases is outdated.) As of 7.0.5, the default behavior is to attempt to match the address with or without the subaddress. If there's a match, then the subaddress will be transferred to any rewritten address. This behavior may be explicitly specified by setting the `subaddressrelaxed` channel option (the default) on

the source channel. `subaddresswild`, if set, will match against subaddresses but disables transfer of the subaddress to the rewritten address. Finally, `subaddressexact` disables special subaddress handling during the reversal process.

33.3.3.29 uucp Option

The legacy configuration `uucp` channel option has been replaced in Unified Configuration by [bangstyle](#).

33.3.3.30 Validating local part of address (`validatelocalnone`, `validatelocalsystem`, `validatelocalexternal`, `validatelocalpopstore`, `validatelocalmsgstore`, `validatelocalprofile`)

The `validatelocal*` channel options control whether any validity check on the local part (username) of an address is performed when messages are enqueued to the channel. Different sorts of channels have different defaults; most channels default to `validatelocalnone`, meaning that no validation of the local part of the address is performed by the channel doing the enqueueing to the channel in question, but the local channel defaults to `validatelocalsystem`, meaning that the local part (username) of an address must be a valid, e-mail receiving account on the system. More specifically, `validatelocalsystem` means that on UNIX platforms, the local part (username) must have an account on the system, or on OpenVMS platforms that the local part (username) must have an account or VMS MAIL profile entry.

When `validatelocalnone` is placed on a channel, messages matching that channel are enqueued to the channel with no validation by the enqueueing channel; it will be up to the destination channel itself to validate the address. So for instance if `validatelocalnone` were placed on the local channel, then incoming SMTP messages apparently matching the local channel would be accepted by the SMTP server and enqueued to the local channel; if the local part turned out not to be a valid account, that would not be discovered until the local channel itself actually ran and checked the local part. (Note that the local channel isn't normally used for actual enqueues in Messaging Server.)

Conversely, if the name space for some other destination channel, say a `MRIF_A1` channel, happened to exactly match the name space for the accounts on the local channel, then placing `validatelocalsystem` on the `MRIF_A1` channel would cause enqueueing PMDF agents such as the SMTP server to reject messages destined for the `MRIF_A1` channel for which the local part (username) could not be validated as if it were a VMS MAIL account.

The `validatelocalexternal`, `validatelocalpopstore`, `validatelocalmsgstore`, `validatelocalprofile` channel options are all currently unimplemented; their behavior is the same as `validatelocalnone`.

33.3.3.31 Require use of aliases (`viaaliasrequired`, `viaaliasoptional`)

The default is `viaaliasoptional`, meaning that an alias match and resulting expansion are not required to be part of address processing prior to enqueueing to this channel. If `viaaliasrequired` is specified on a channel, then only addresses whose processing involved expansion of some sort of alias ([LDAP entry](#), [alias file entry](#), [alias database entry](#), *etc.*) are allowed to be enqueued to the channel.

Note that `viaaliasrequired` must be used on the local channel for [direct LDAP configuration](#) to work properly.

33.3.4 Attachments and MIME processing channel options

A number of channel options affect the processing of so-called attachments and MIME parts.

See also the [conditionalpassthrough](#) and [conditionalrelay](#) channel options which, by setting operation type, have implications regarding "fixup" of MIME structure in messages, and also the [inner and noinner](#) channel options which, by controlling whether the MTA's processing looks "inside" messages, among other things affects "fixup" of the MIME header lines on encapsulated parts of messages. See also the [limitheadertermination](#) and [relaxheadertermination](#) channel options which, by controlling what is interpreted as the division between the message header and the message body, thereby affects detection of a message's MIME structure.

33.3.4.1 Processing within security multiparts (`conditionalsecuritymultiparts`, `processecuritymultiparts`, `retainsecuritymultiparts`)

The `conditionalsecuritymultiparts`, `processecuritymultiparts`, and `retainsecuritymultiparts` channel options control the handling of security multiparts, (that is, multipart/signed and multipart/encrypted parts), by the MTA's message structure parsing code. They are particularly relevant for the [conversion channel](#), as they control whether the conversion channel processes "inside" such multiparts. `retainsecuritymultiparts` is the default: multipart/signed and multipart/encrypted are treated as monolithic and not "looked inside" by MIME message structure parsing. For instance, by default (`returnsecuritymultiparts`) the conversion channel won't process any parts inside such parts. With `processecuritymultiparts`, the conversion channel sees the parts inside the security multipart; note that enabling this breaks all signatures, since the MTA redoes all the boundary markers during its MIME structure parsing. `conditionalsecuritymultiparts` processes the multiparts similarly to `processecuritymultiparts`, but retaining any "preamble" material inside the multipart.

33.3.4.2 `convert_octet_stream` Option

The legacy configuration `convert_octet_stream` channel option has been replaced in Unified Configuration by [`convertoctetstream`](#).

33.3.4.3 Conversion of application/octet-stream material (`convertoctetstream`, `noconvertoctetstream`)

MIME provides a general-purpose type for exchange of pure untyped binary data. Such data may or may not be usable in any given circumstance; no other information about the data is available. Various MTA channels provide mechanisms for dealing with such data that may or may not be appropriate. The `convertoctetstream` and `noconvertoctetstream` options control these mechanisms; if the former is specified on a source channel conversions are performed and if the latter is specified no conversions are performed. The latter option is the default for all channels.

`convertoctetstream` is not relevant for any currently available Oracle Messaging Server channels.

33.3.4.4 Automatic defragmentation of message/partial messages (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. Information is included in each part so that the message can be automatically reassembled once it arrives at its destination.

The `defragment` channel option and the [defragmentation channel](#) provide the means to reassemble messages in the MTA. When a channel is marked `defragment` any message/partial messages queued to the channel will be placed in the defragmentation channel queue instead. Once all the parts have arrived the message is rebuilt and sent on its way.

The `nodefragment` disables this special processing. `nodefragment` is the default.

A `defragment` channel must be present in the configuration in order for the `defragment` channel option to have any effect. Initial configuration normally includes a `defragment` channel in the MTA configuration.

33.3.4.5 Encoding header interpretation (ignoreencoding, ignoremessageencoding, ignoremultipartencoding, interpretencoding, interpretmessageencoding, interpretmultipartencoding)

The MTA has the ability to convert various non-standard message formats to MIME via the Yes [CHARSET-CONVERSION](#). In particular, the [RFC 1154](#) format uses a non-standard 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. The `ignoreencoding` source channel option instructs the MTA to ignore any Encoding: header. (Note that unless the MTA has a `CHARSET-CONVERSION` enabled, such headers will be ignored in any case.) The `interpretencoding` source channel option instructs the MTA to pay attention to any Encoding: header line, if otherwise configured to do so, and is the default.

New in 6.3 are the `ignoremessageencoding`, `interpretmessageencoding`, `ignoremultipartencoding`, and `interpretmultipartencoding` source channel options. The MIME standards, [RFC 2045](#) (which updates [RFC 1521](#)) and [RFC 2046](#), restrict the set of allowed content-transfer-encodings permitted on MIME multipart or message parts to 7BIT, 8BIT, or BINARY in general, with the particular message subtypes message/partial and message/external-body being further restricted to allow only 7BIT. Nevertheless, buggy/incompliant software may sometimes emit messages that illegally label multipart or message parts as having another content-transfer-encoding. Now when such an illegal encoding label is seen, the question is whether the material is in fact encoded (illegally) as claimed, or whether the material is not in fact encoded and the claim is simply false.

Prior to 7.0.5, the MTA's default handling (and the only handling available prior to 6.3) is to believe the Content-transfer-encoding: label--and the MTA can (and will) "decode" such messages. This corresponds to `interpretmessageencoding` and `interpretmultipartencoding` channel options. This leads to "successful" handling of messages that are broken due to illegally being encoded--but will not be equally satisfactory

for messages that are broken due to an outright false labelling with the contents not actually being encoded.

Alternatively, the new `ignoremessageencoding` and `ignoremultipartencoding` channel options, when placed on a source channel, will cause the MTA to ignore any claimed Content-transfer-encoding: on message parts or multipart parts, respectively, which can be more useful when broken software is emitting messages that falsely claim encoding of such parts. (Note that since what [RFC 2045](#) specifies as the permitted and legal 7BIT, 8BIT, and BINARY content-transfer-encodings are all identity encodings---no transformation of the data is involved, with the content-transfer-encoding label merely recording what sort of material the part contains, which the MTA can determine for itself---it causes no harm to [RFC 2045](#) conformant message or multipart parts to ignore the content-transfer-encoding. So for the moment, use of `ignore*encoding` channel options is "safe" for [RFC 2045](#) legal messages. **However**, the experimental EAI (Email Address Internationalization) specifications are likely to change this MIME restriction, permitting encodings on such parts; at such time, support for correct messages would require respecting and interpreting any encodings on such parts. So the `ignore*encoding` channel options, while a useful and safe-for-the-moment short-term workaround for broken, remote software, must be considered short-term: they may not continue to be safe to use in future.)

Note that the `imsimta test -mime` utility has switches for testing and describing the (MIME) structure of messages, switches which correspond to these channel keywords.

Note that in Messaging Server 7.0u3 and prior versions, the `ignoremultipartencoding` and `ignoremessageencoding` channel options would have no effect (be ignored) when placed on a [conversion](#) or SMS channel, or on any site-written channels.

The `interpretmessageencoding` channel option is the default in all versions. Prior to 7.0.5, `interpretmultipartencoding` was also the default. But as of 7.0.5, `ignoremultipartencoding` is the default.

33.3.4.6 Soft wrap (encode) long lines in messages (`linelength`)

The SMTP specification allows for lines of text containing up to 1000 bytes. However, some transports may impose more severe restrictions on line length, and even some SMTP systems, in violation of the relevant standards, cannot handle full length lines.

First the MTA performs any appropriate header line wrapping: see the [headerlinelength](#) channel option. Then the `linelength` channel option provides a mechanism for limiting the maximum permissible message body line length on a channel by channel basis. Messages queued to a given channel with body lines longer than the `linelength` limit specified for that channel will have the message body encoded automatically. (Note that `linelength` is a destination channel option modifying what the MTA *emits*; for controlling how the MTA handles illegally long lines that it *receives* via SMTP, see instead the [*smtpplonglines](#) options.) The various encodings available 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 appropriating decoding filter. (In most cases MIME-aware mail user agents are able to detect that such decoding is necessary and perform it automatically.)

Note that encoding can only reduce line lengths to fewer than 80 characters. For this reason specification of line length values less than 80 may not actually produce lines with lengths that comply with the stated restriction.

Note also that `linelength` causes encoding of data so as to do "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 instead the "Record,Text" [CHARSET-CONVERSION](#).

The default for arbitrary channels is 1023 channels; but channels marked with an [smtp*](#) or [lmtp*](#) channel option will not allow more than 998 characters and attempts to set `linelength` larger (or not setting `linelength` explicitly on such channels) will result in using 998 characters on such channels.

33.3.4.7 Automatic fragmentation of large messages (`maxblocks`, `maxlines`)

Some mail systems or network transports 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 will automatically be 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.

Message fragmentation and defragmentation may also be used to effectively provide "checkpointing" of message transmission.

The `maxblocks` and `maxlines` channel options are used to impose size limits beyond which automatic fragmentation will be activated. Both of these channel options require a single integer argument. `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` MTA option. `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. Since message headers cannot be split into multiple messages, and yet they themselves may 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` MTA options.

`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` channel option). If the message header is larger, the MTA takes the product of `max_header_block_use` and `maxblocks` as the size of the header; *i.e.*, 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 that is larger than 5 blocks is treated as a 5 block header, and if the message is 5 or fewer blocks in size it will not be fragmented. A value of 0 will cause 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. Note, however, that each fragment will always contain at least one message line, regardless of whether or not the limits are exceeded by this.

The `max_header_line_use` channel option operates in a similar fashion in conjunction with the `maxlines` channel option.

See the [defragment](#) channel option and the [Defragmentation channel](#) for discussion of the reverse operation: that is, how the MTA can be configured to perform automatic defragmentation of message fragments that it *receives*.

33.3.4.8 Microsoft Exchange gateway channels (`msexchange`, `nomsexchange`)

The `msexchange` channel option may be used on [TCP/IP channels](#) to tell the MTA that this is a channel that communicates with Microsoft[®] Exchange gateways and clients. Use of the option tells the MTA to try and accommodate nonstandard behavior on the part of Microsoft Exchange. Exactly what nonstandard behaviors are dealt with is subject to change.

Currently the `msexchange` channel option on a channel configured to allow TLS use (see the [*tls* channel options](#)) causes advertisement (by the MTA's SMTP server) and recognition (by the MTA's SMTP client) of the non-standard TLS capability string, in addition to the standard STARTTLS capability string, to indicate that TLS is supported.

New in 7.0.5, setting `msexchange` on a destination channel will cause the MTA, if performing any sort of MIME processing operation, to remove any Content-disposition: header line from any text/calendar message parts, as despite Content-disposition:'s long-standing existence as a standardized header line, not to mention the basic MIME rule that unrecognized Content-* header lines should be ignored, Microsoft[®] Outlook's handling of text/calendar parts is disturbed when such parts have a Content-disposition: specified. So specifying `msexchange` on a channel sending to Microsoft Exchange, if text/calendar parts will flow through that channel, should allow Microsoft Outlook to process calendar parts more successfully.

`nomsexchange` is the default.

33.3.4.9 MIME Content-type: and Content-disposition: header line parameter lengths (`nameparameterlengthlimit`, `parameterlengthlimit`)

A number of popular e-mail clients (mail user agents) have had a history of security problems involving buffer overruns during header line processing, such as during processing of the Content-type: or Content-disposition: header lines. So although [RFC 2045 \(MIME\)](#) does not impose any length constraints on such parameters, keeping in mind the historical vulnerabilities of many popular e-mail clients, the MTA normally truncates MIME parameters in an attempt to protect any downstream, vulnerable clients. By default, the MTA truncates the Content-type: NAME and Content-disposition: FILENAME parameters at 128 characters each, and other general parameters at 1024 characters. These defaults may be changed by using the `nameparameterlengthlimit` and `parameterlengthlimit` channel options, respectively. Each takes an integer argument specifying the desired maximum length to allow for such parameters; (longer parameters will be truncated).

A distinct but related issue to parameter truncation is parameter segmentation, per [RFC 2231](#) rules, which can be controlled in part by the `parameterformatdefault` and related channel options.

33.3.4.10 `noconvert_octet_stream` Option

The legacy configuration `noconvert_octet_stream` channel option has been replaced in Unified Configuration by `noconvertoctetstream`.

33.3.4.11 Convert some non-standard "attachments" to MIME format (`thurman`, `nothurman`, `uma`, `nouma`)

The `thurman` channel option on a source channel causes the MTA to "sniff" incoming, non-MIME messages and convert them to MIME format, in the process converting certain non-standard "attachment" formats (e.g., uuencoded or BinHex "blobs" embedded in messages) into MIME format attachments. While this may seem like a positive transformation to perform on messages, modifications of message content should always be a last resort, the resulting MIME-ification of messages has limitations and may be considered undesirable in some circumstances or to some users, this "sniffing" does incur processing overhead, and such an approach may appear as a deceptively "easy" fix for communicating with non-standard e-mail software (when a better, more permanent solution would be to upgrade the out-of-date, non-standard software that is emitting such "blobs"). As such, the `nothurman` channel option is the default -- and any decision to configure with `thurman` should be made with due consideration and caution.

The `uma` source channel option is like `thurman`, except that with `uma`, the message "sniffing" and content transformation only occurs if the MTA was already performing message body processing (and hence would have been converting the message body to MIME format already), for instance cases of incoming non-MIME messages that illegally contain unlabelled eight bit characters. `nouma` is the default.

Note that [CHARSET-CONVERSION mapping table](#) keywords exist to configure `thurman` or `uma` type processing on a somewhat more selective basis.

The `thurman` or `uma` processing on its own merely causes "blobs" in non-MIME messages to get converted to fairly "generic" MIME parts, with Content-type of `application/octet-stream` and a Content-transfer-encoding of `X-UUENCODE`. Optionally, one may configure to guess (typically based on filename, especially filename extension) at a better Content-type labelling, and/or to convert to a different Content-transfer-encoding, using the MTA's facility for [Relabelling MIME header lines](#).

33.3.4.12 MIME Content-type: and Content-disposition: header line parameter RFC 2231 encoding (parameterformatdefault, parameterformatminimizeencoded, parameterformatstripencoded)

As of Messaging Server 7.2-0.01, the MTA supports [RFC 2231 \(MIME Parameter Value and Encoded Word Extensions\)](#), thus supporting use of alternate character sets and languages in MIME parameters, as well as supporting the segmentation of "long" parameter values.

As a separate issue from the truncation of very long parameter values as controlled by [parameterlengthlimit](#) and [nameparameterlengthlimit](#), note that as of Messaging Server 7.2-0.01 when [RFC 2231](#) support was added, the MTA will automatically segment long parameter values according to [RFC 2231](#) rules. (Note that the length at which [RFC 2231](#) segmentation is triggered is not configurable.) For Messaging Server 7.2-0.01, parameter values over 65 characters in length will automatically be segmented into 40 character segments; e.g.,

```
filename="veryveryveryveryveryveryveryveryveryveryveryveryveryveryveryverylong.name"
```

would become

```
filename*0="veryveryveryveryveryveryveryveryveryvery";
```

```
filename*1="veryveryveryveryverylong.name"
```

As of Messaging Server 7.4-0.01 and the implementation of CR # 6924445, the length limit for triggering parameter segmentation was increased from the prior 65 characters up to 70 characters. (In particular, as [RFC 2046](#) limits the length of MIME boundary delimiters to at most 70 characters, this larger trigger length avoids triggering MIME parameter segmentation of compliant boundary delimiters.)

New in 7.0.5 are the channel options `parameterformatdefault`, `parameterformatminimizeencoded`, and `parameterformatstripencoded`, with these last two options providing new features to aide with cases of dealing with other software that does not yet support [RFC 2231](#). `parameterformatdefault` is the default and means to do normal [RFC 2231](#) encoding, as needed. `parameterformatminimizeencoded` tells the MTA to attempt to remove any unnecessary or redundant [RFC 2231](#) encoding from MIME parameters including removing any [RFC 2231](#) segmentation of parameters; when it is applied, parameter segmentation will be removed and those encoded words not involving charset or language information or 8 bit characters will be replaced with regular parameter values. `parameterformatstripencoded` tells the MTA to strip any characters that would require [RFC 2231](#) encoding from MIME parameters, thereby allowing the parameter to be represented without any [RFC 2231](#) encoding or segmentation.

For webmail (MSHTTP) generation of [RFC 2231](#) encoded format, see the [rfc2231compliant](#) MSHTTP option.

33.3.5 BSMTP-specific channel options

A few (now DEPRECATED or RESTRICTED) channel options relate to BSMTP-specific protocol details. For channel options relevant to today's [BSMTP channels](#), see instead more general options such as the [SMTP and LMTP protocol channel options](#) and the [serviceconversion](#) channel option.

33.3.5.1 Batch SMTP Continuation Lines (`contchar`, `contposition`)

DEPRECATED.

The `contposition` channel option specifies the point at which batch SMTP lines are folded onto a continuation line. The default value for `contposition` is 0, which disables continuation lines.

The `contchar` channel option is used to specify a continuation character for commands in batch SMTP. Commands longer than `contposition` characters long will have the character specified by the integer argument to `contchar` inserted in the `contposition` position and the remainder of the line will be wrapped to another line. The default value for `contchar` is 0.

See also the TCP/IP-channel-specific option [CONTINUATION_CHARS](#) which allows for specification of additional continuation characters.

These options were used to accomodate the peculiar form of batch SMTP that was employed by BITNET. Now that BITNET is no more these options are obsolete and deprecated.

33.3.5.2 Generation of TICKet BSMTP Commands (`tick`, `notick`)

Some batch SMTP (BSMTP) implementations require the presence of a ticket number, specified with the `TICK BSMTP` command. The `tick` channel option tells the MTA to issue this command; `notick` suppresses it. `tick` is the default on channels that support tickets. Currently only PMDF's `BN_MASTER` channel program uses this channel option.

33.3.5.3 Generation of VERbose BSMTP Commands (`verb_on`, `verb_off`, `verb_none`, `verb_never`)

Some batch SMTP (BSMTP) implementations support the use of the `VERB` command to control the nature of their replies. On the client side the `verb_on` command tells the MTA to issue a `VERB ON` command in the BSMTP command sequence; `verb_off` tells the MTA to issue a `VERB OFF` command. The `verb_never` and/or the `verb_none` channel option tells the MTA not to issue any `VERB` commands. `verb_never` is the default, and this default should *not* be changed.

On the server side, the `verb_never` channel option causes `VERB` commands in the BSMTP stream to be accepted but ignored. `verb_none` can be used to enable processing of `VERB` commands.

33.3.6 Character sets and eight bit data channel options

A number of channel options control handling of character sets in messages, as well as eight bit data.

See also the general topic of [Character set conversion](#).

33.3.6.1 Automatic character set labelling (`charset7`, `charset8`, `charsetesc`)

The MIME specification provides a mechanism to label the charset used in plain text messages: A "charset=" parameter can be specified as part of the Content-type: header line. Various charset names are defined in MIME, including US-ASCII (the default), ISO-8859-1, ISO-8859-2, and so on, and many more have been registered with the Internet Assigned Numbers Authority (IANA).

Some existing systems and user agents, however, do not provide any mechanism for generating these charset labels. The `charset7`, `charset8` and `charsetesc` channel options, when placed on a source channel, provide a mechanism to specify charset names to be inserted into message headers. Each option requires an argument giving a charset name. The names are not checked for validity. Note, however, that [charset conversion](#) can only be done on charsets specified in the MTA's charset definition file `charsets.txt`. The names defined in this file should be used if possible.

The `charset7` charset name is used if the message contains only seven bit characters; the `charset8` name will be used if eight bit data is found in the message; `charsetesc` will be used if a message containing only seven bit data happens to contain one or more escape characters. If the appropriate option is not specified no character set name will be inserted during MIME processing into Content-type: header lines for text parts that lack an existing charset label.

When the presence of a `charset*` channel option on a channel causes a MIME "charset" parameter clause to be added to an incoming message, that of course also means that the

message gets the more fundamental MIME-version: and Content-type: header lines added, if not already present.

New in Messaging Server 7.4-18.01, the `charset7`, `charset8`, and `charsetesc` channel options will also cause labelling (with the specified charset) of incoming illegal, unlabelled parameter values on MIME Content-type: or Content-Disposition: header lines, when such parameters must be encoded due to their content. That is, while such parameter values have always been subject to encoding (to make them syntactically legal, if not necessarily usable) by the MTA, the new feature is that charset labelling will be inserted also, making them more usable.

Note that the `charset8` option also controls the MIME encoding of eight bit characters found in message headers (where such eight bit data is unconditionally illegal). The MTA will normally always MIME encode any such (illegal) eight bit data encountered in message headers, labelling it as the UNKNOWN charset if no `charset8` value has been specified on the current source channel. (Actual addresses are a special case. In the actual address, that is, in the [RFC 822](#) addr-spec, where eight bit categorically must not appear, any eight bit data will be replaced by the MTA with the asterisk character, *. Note that an [RFC 822](#) phrase, or "personal name", however, is subject to the above described MIME encoding of any illegal eight bit, using the `charset8` charset name.)

These charset specifications never override existing labels; that is, they have no effect if a message already has a charset label or is of a type other than text.

The `charsetesc` option tends to be particularly useful on channels that receive unlabelled messages using Japanese or Korean character sets that contain the escape character (*e.g.*, `iso-2022-jp` or `iso-2022-kr`).

33.3.6.2 Eight bit SMTP capability and EAI capability (`eightbit`, `eightnegotiate`, `eightstrict`, `sevenbit`, `utf8header`, `utf8negotiate`, `utf8strict`)

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. Indeed, there have even been past cases of SMTP servers which will crash when presented with eight bit data.

The MTA 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 on a given channel by specifying the `sevenbit` channel option. A channel should be marked `eightbit` if no such restriction exists.

Some transports such as extended SMTP may actually support a form of negotiation to determine if eight bit characters can be transmitted. The `eightnegotiate` channel option can be used 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` source channel option tells the MTA to reject any messages that contain unnegotiated eight bit data; the exact text of this error may be controlled via the `error_text_unnegotiated_eightbit` MTA option.

33.3.6.3 Unencoded non-ASCII headers (`headerset7`, `headerset8`, `headersetesc`)

In extremely rare situations very old and standards-incompliant SMTP servers must be accommodated. One of the behaviors such servers sometimes exhibit is an inability to deal with MIME encoded words in headers. Instead such servers expect headers to simply contain unencoded material in some other charset.

The `headerset7`, `headerset8`, and `headersetesc` channel options are used to deal with such situations. Each accepts a charset name as an argument. When applied to a destination channel, they cause encoded words in the specified charsets to be decoded. Any combination of the options can be specified, meaning from 1 to 3 charsets can be decoded. Note that the names of these options were selected to match up with other [charset](#) options but there is essentially no difference between the three options.

Extreme care should be exercised when using these options, as the messages they produce will be grossly standards incompliant and may cause serious interoperability problems, to the point of crashing some very old SMTP servers.

33.3.6.4 MIME Content-type: and Content-disposition: header line parameter RFC 2231 encoding (`parameterformatdefault`, `parameterformatminimizeencoded`, `parameterformatstripencoded`)

As of Messaging Server 7.2-0.01, the MTA supports [RFC 2231 \(MIME Parameter Value and Encoded Word Extensions\)](#), thus supporting use of alternate character sets and languages in MIME parameters, as well as supporting the segmentation of "long" parameter values.

As a separate issue from the truncation of very long parameter values as controlled by [parameterlengthlimit](#) and [nameparameterlengthlimit](#), note that as of Messaging Server 7.2-0.01 when [RFC 2231](#) support was added, the MTA will automatically segment long parameter values according to [RFC 2231](#) rules. (Note that the length at which [RFC 2231](#) segmentation is triggered is not configurable.) For Messaging Server 7.2-0.01, parameter values over 65 characters in length will automatically be segmented into 40 character segments; *e.g.*,

```
filename="veryveryveryveryveryveryveryveryveryveryveryveryveryveryveryveryverylong.name"
```

would become

```
filename*0="veryveryveryveryveryveryveryveryveryvery";  
filename*1="veryveryveryveryveryverylong.name"
```

As of Messaging Server 7.4-0.01 and the implementation of CR # 6924445, the length limit for triggering parameter segmentation was increased from the prior 65 characters up to 70 characters. (In particular, as [RFC 2046](#) limits the length of MIME boundary delimiters to at most 70 characters, this larger trigger length avoids triggering MIME parameter segmentation of compliant boundary delimiters.)

New in 7.0.5 are the channel options `parameterformatdefault`, `parameterformatminimizeencoded`, and `parameterformatstripencoded`, with these last two options providing new features to aide with cases of dealing with other software that does not yet support [RFC 2231](#). `parameterformatdefault` is the default and means to do normal [RFC 2231](#) encoding, as needed. `parameterformatminimizeencoded` tells the MTA to attempt to remove any unnecessary or redundant [RFC 2231](#) encoding from MIME

parameters including removing any [RFC 2231](#) segmentation of parameters; when it is applied, parameter segmentation will be removed and those encoded words not involving charset or language information or 8 bit characters will be replaced with regular parameter values. `parameterformatstripencoded` tells the MTA to strip any characters that would require [RFC 2231](#) encoding from MIME parameters, thereby allowing the parameter to be represented without any [RFC 2231](#) encoding or segmentation.

For webmail (MSHTTP) generation of [RFC 2231](#) encoded format, see the `rfc2231compliant` MSHTTP option.

33.3.7 Conversion tag and service conversion channel options

One place at which [conversion tags](#) can be added is on a destination or source channel basis, controlled by channel options.

Triggering the check for service conversions can be controlled by channel options.

33.3.7.1 Channel-based conversion tags (`destinationconversiontag`, `sourceconversiontag`)

(New in 7.0.5.) The channel options `destinationconversiontag` and `sourceconversiontag` allow for per-channel addition of [conversion tags](#) to messages. Each option accepts a single argument consisting of a comma-separated list of conversion tags. The source conversion tags are attached to all message recipients; any destination conversion tags are attached to all recipients associated with that destination channel.

See also the domain level and user level LDAP attributes for adding per-sender and per-recipient conversion tags, `ldap_domain_attr_source_conversion_tag`, `ldap_domain_attr_conversion_tag`, `ldap_source_conversion_tag`, and `ldap_conversion_tag`, and in Unified Configuration the alias option `alias_conversion_tag` or in legacy configuration the `[CONVERSION_TAG] alias file named parameter`.

33.3.7.2 Source channel trigger for service conversions (`serviceconversion`, `noserviceconversion`)

The `serviceconversion` and `noserviceconversion` channel options are new under these names in Messaging Server 7.3-11.01; in earlier versions, the names (obsolete for Unified Configuration) `service` and `noservice` had been used.

Instead of triggering a check for applicable [service conversions](#) via a matching entry in the [CHARSET-CONVERSION mapping table](#), setting `serviceconversion` on a source channel equivalently triggers the check for applicable service conversions, for all messages coming in that channel. `noserviceconversion` is the default, and means that the `CHARSET-CONVERSION` mapping table is, as normal, the trigger for whether to check for applicable service conversions.

33.3.8 Display label channel options

A couple of channel options are intended for labelling purposes, for future use.

33.3.8.1 Channel caption and description fields (`caption`, `description`)

The `description` channel option takes a string argument and provides a way to associate a descriptive term or phrase with a channel. This feature is intended for future management utility use. The (new in 6.3) `caption` channel option is similar, though it would normally be given a shorter argument, one suitable for use as the caption for a table column, for instance.

33.3.9 DKIM channel options

A number of channel options affect DKIM (DomainKeyws Identified MAIL) processing. See also the [DKIM MTA options](#).

33.3.9.1 DKIM channel options: (`destinationdkimignore`, `destinationdkimpreserve`, `destinationdkimremove`)

The `destinationdkim*` channel options are destination channel analogues of the `dkim*` channel options. They operate in the same sort of way as their source channel analogues, except that note that since destination channel determination is made later in message processing, after *some* message processing has already occurred, then for instance the switch to "`passthrough`" mode resulting from `destinationdkimpreserve` applying will occur *after* some message processing may already have occurred. Thus `destinationdkimpreserve` may be more specific in terms of which messages it applies to, but is potentially less comprehensive in its avoidance of message processing, than `dkimpreserve`.

33.3.9.2 Source channel handling of DKIM-Signature: header fields (`dkimignore`, `dkimpreserve`, `dkimremove`)

DKIM-Signature: header fields may require special handling, or their presence may indicate the need for special handling. The `dkimignore`, `dkimpreserve`, and `dkimremove` source channel options provide various capabilities in this area.

The `dkimignore` source channel option instructs the MTA to take no special action in regards to DKIM-Signature: header fields. This option is the default.

The behavior of the `dkimpreserve` source channel option depends on whether the `dkim_preserve_domains` and `dkim_ignore_domains` MTA options are set. If neither of these options are set, the presence of any DKIM-Signature: header field in the message puts the MTA in "`passthrough`" mode, where no header rewriting will be performed.

If either of the `dkim_ignore_domains` or `dkim_preserve_domains` MTA options is set, every DKIM-Signature: header field that is present will be parsed and the domain value specified by the "d=" will be extracted. Each extracted domain is first compared against the space-separated list of domains specified by the `dkim_ignore_domains` MTA option. If a match is found no action is taken and processing continues with the next DKIM-Signature: field. If no match is found the domain is next checked against the space-separated list of domains specified by the `dkim_preserve_domains` MTA option. If a match is found there the MTA is placed in "`passthrough`" mode and scanning is terminated.

The behavior of the `dkimremove` source channel option depends on whether the `dkim_remove_domains` and `dkim_ignore_domains` MTA options are set. If neither of these options are set, all DKIM-Signature: fields are unconditionally removed from the message.

If either of the `dkim_remove_domains` or `dkim_ignore_domains` MTA options is set, every DKIM-Signature: header field that is present will be parsed and the domain value specified by the "d=" will be extracted. Each extracted domain is first compared against the space-separated list of domains specified by the `dkim_ignore_domains` MTA option. If a match is found no action is taken and processing continues with the next DKIM-Signature: field. If no match is found the domain is next checked against the space-separated list of domains specified by the `dkim_remove_domains` MTA option. If a match is found there the corresponding DKIM-Signature: field is removed from the message.

See also the analogous destination channel options `destinationdkim*`, introduced in 8.0.

33.3.10 Error interpretation channel options

A couple of channel options override general [MTA option settings regarding error interpretation](#).

33.3.10.1 Error interpretation (`usepermanenterror`, `usetemporaryerror`)

The `usepermanenterror` and `usetemporaryerror` source channel options override, on a per-source-channel basis, the `use_permanent_error` and `use_temporary_error` MTA options, respectively.

33.3.11 File creation in the MTA queue area channel options

Several channel options can affect the creation of message files in the MTA queue area.

IMPORTANT NOTE: The various MTA queue directories are reserved for use by Oracle software only. Customers MUST NOT create files in these areas.

33.3.11.1 Addresses per message copy (`multiple`, `addrspersfile`, `single`, `single_sys`)

The MTA allows multiple destination addresses to appear in each queued message copy. Some channel programs, however, may only be able to process messages with one recipient per copy, or with a limited number of recipients, or with a single destination system per message copy. For example, the SMTP client programs for 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 TCP/IP channel is typically used for all outbound Internet message 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 handle errors in this area at all gracefully.

The channel options `multiple`, `addrspersfile`, `single`, and `single_sys` can be used to control how the MTA handles multiple addresses. `single` means that a separate copy of the message should be created for each destination address on the channel. `single_sys` creates a single copy of the message for each destination system (more precisely, each destination domain name) associated with a recipient address. `multiple` creates a single copy of the message for the entire channel. Note that at least one copy of each message is created for each channel the message is queued to, regardless of the options used. `multiple` is the default

for all channels marked with the `nosmtp` channel option. Channels marked with one of the `smtp*` channel options default to `single_sys` unless the `daemon` option has been set. Prior to 7.0 channels marked with one of the LMTP options defaulted to `multiple`, as of 7.0 they also default to `single_sys` unless `daemon` has been set. (Actually, calling this the "default" is something of a misnomer - without `daemon` SMTP and LMTP channels will be forced to `single_sys` regardless of the channel setting. This is done to prevent delivery of messages to the wrong system and possibly to the wrong user.)

These options also affect the **Job Controller's** "sorting" and organization of messages on a channel. The `single_sys` option causes the Job Controller to organize messages into separate internal lists based on destination domain. (And hence a command such as `imsimta qm messages` will show messages sorted by destination host.)

The `addrspersfile` channel option is used to put a limit on the maximum number of recipients that can be associated with a single message file in an MTA channel queue, thus limiting the number of recipients that will be processed in a single operation. This option requires a single integer argument specifying the maximum number of recipient addresses allowed in a message file; if this number is reached the MTA will automatically create additional message files to accommodate them.

Note that the default of `addrspersfile` depends on the channel type. For most channel types, `addrspersfile` defaults to effectively unlimited (2147483647), but setting an `smtp*` option on a channel causes `addrspersfile` to default to 99, while setting an `lmtp*` option on a channel causes `addrspersfile` to default to 999.

The default messages per channel copy setting varies by channel type. The `multiple` option is the default for everything except `tcp_*` channels. `tcp_*` channels default to `multiple` if `daemon` is set, and `single_sys` if it is not. The combination of an explicit `multiple` and no `daemon` isn't allowed on `tcp_*` channels and will be overridden by forcing a setting of `single_sys`.

33.3.11.2 Expansion of multiple addresses (`expandlimit`, `expandchannel`, `holdlimit`)

Most MTA 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 ("on-line" delays). In particular, multi-recipient messages that require a great deal of processing of the message body can be affected by processing delays, or that require creation of many different file copies on disk in the MTA queue area can be affected by slow disk performance. If the resulting delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred ("off-line") processing of additional recipient message copies once a given number of addresses are specified for a single message. The deferral happens after processing of the "initial" recipients (those before the `expandlimit` value was reached), and after address processing for the additional recipients too, (for instance, after `*_ACCESS mapping table` checks and after alias processing), but before message processing. In particular, such deferral means that for the "additional" recipients, only *one* message file (storing all of the "additional" recipients), is written to the queue area (to a `reprocess*` or `process*` channel queue area, depending upon use of the `expandchannel` channel option). 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` channel option. The `expandlimit` option takes an integer argument that specifies at what number of recipients to begin deferring processing of the message copy (or copies) to that and additional recipient addresses. The default value is effectively infinite if the `expandlimit` channel option is not specified. A value of 1 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` channel option 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` channel option; 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 some purposes. In particular, for Sun ONE Messaging Server MTA versions 5.2 and earlier, typical configuration usage required that a processing channel, rather than a reprocessing channel, be used. 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` option to have any effect. If your configuration was built by the initial configuration utility, then you should already have a reprocessing channel.

(Note that typical Messaging Server sites running version 5.2 or earlier could not use the `expandlimit` option unless they also marked the affected channel `expandchannel process` (or `expandchannel process_somethingorother` redirecting the expansion to an alternate `process_*` sort of channel), as enqueues to a channel marked `viaaliasrequired` would not succeed if deferred to a `reprocess*` channel.)

Extraordinarily large lists of recipient addresses are often a characteristic of so-called SPAM--junk e-mail. The `holdlimit` channel option 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` channel option). As `.HELD` messages, the files will sit unprocessed in that MTA queue area awaiting manual intervention by the MTA postmaster.

33.3.11.3 Using multiple subdirectories to store queued messages (subdirs)

The MTA by default spreads all messages queued to a channel across 20 subdirectories. However, a channel which handles a large number of messages and tends to build up a large store of message files waiting for processing, *e.g.*, a TCP/IP channel, may get better performance out of the file system if those message files are spread across even more subdirectories. The `subdirs` channel option provides this capability: it accepts an integer argument which specifies the number of subdirectories across which to spread messages for the channel. The allowed range of integer values is 1 through 999. The default value is 20.

To disable the use of subdirectories for channel queues, use the `nosubdirs` keyword.

33.3.12 Gateway or firewall or mailhub channel options

There are a number of channel options of particular interest when configuring channels intended to communicate with gateway, firewall, or mailhub systems.

33.3.12.1 Force "detour" routing of hosted users (`aliasdetourhost`, `aliasoptindetourhost`)

The (new in iMS 5.2p2, and JES MS 6.1) `aliasdetourhost` channel option allows source-channel-specific overriding of hosted users' `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. It allows better configuration and use of "intermediate filtering" sorts of channels and third party filtering hosts.

The `aliasdetourhost` channel option takes a single host/domain name as an argument. When specified on a source channel, this channel option causes alias expansion of addresses stored in LDAP to stop (short-circuit) just prior to the point where `mailHost` (more precisely, the attribute named by the `ldap_mailhost` MTA option) information is checked. The host specified by the `aliasdetourhost` channel option is used as the (assumed to be non-local) `mailHost`. That is, a source route containing the specified host is added to the address (just as if a non-local `mailHost` had been found) and processing continues onward from that point. Note that in particular, this forced use of the `aliasdetourhost` specified host as a non-local `mailHost` stops further expansion of the alias for purposes of things such as application of user forwarding and Sieve filter application (which normally would occur subsequently during alias expansion when a user's real `mailHost` is this MTA).

Thus use of `aliasdetourhost` on an incoming channel lets the MTA do address validation (check that an incoming address corresponds to a valid user entry), while "delaying" complete expansion and processing (in particular, forwarding and Sieve evaluation) of the valid local recipient addresses. This combination of effects is potentially very useful.

A typical application of this channel option is for purposes of "detouring" messages through a special channel or host, most often for purposes of spam/virus filtering. It is often used in conjunction with use of an "alternate" conversion channel for such "detour" purposes, where the "alternate" conversion channel approach is used to handle cases of non-local recipient addresses, while `aliasdetourhost` is used to handle cases of local-to-this-`mailHost` recipient addresses. (Use of an "alternate" conversion channel approach for a routing "detour" on local-to-this-`mailHost` recipient addresses incurs various problems, in particular in the areas of forwarding and Sieve filter evaluation timing. It is desirable to delay Sieve filter evaluation until after the "detour" - for instance, so that Sieve filters can look for headers added by the "detour" host. It is also desirable to delay application of user forwarding until after the "detour", to avoid potential duplication of the forwarding. Such a delay in the final parts of user alias expansion is exactly what `aliasdetourhost` can be used to achieve.)

The (new in JES MS 6.2p4) `aliasoptindetourhost` option has the same function as `aliasdetourhost`, except that it only applies for users in LDAP who have "opted-in" via whatever user attribute is named by the `ldap_detourhost_optin` MTA option, or whatever domain attribute is named by the `ldap_domain_attr_detourhostoptin` MTA option. The argument of the `aliasoptindetourhost` channel option specifies a list of detour hosts separated by commas. The value(s) of the `optin` attribute are compared with the list; the first match will be used as the "override" `mailHost` for any users who are "opted-in". However, any attribute that doesn't contain at least one period (which would be necessary to match a legitimate mail host) is treated as an effective wildcard; the first host from the list will be used in this case.

Finally, if the option value matches the special value specified by the `aliasdetourhost_null_optin` MTA option it will simply be ignored. This mechanism is provided to accomodate provisioning systems that insist on every known attribute having

a value. Omitting the attribute value entirely is the preferred method for disabling detour processing, however.

33.3.12.2 Clone messages to alternate destination (`clonehosts`)

A commonly requested capability is to "clone" all messages that meet some criteria and send them to an alternate destination. This can be accomplished in a variety of ways, including the Sieve ["capture" action](#), the [MESSAGE-SAVE-COPY mapping](#), the [FORWARD mapping](#), and various sorts of address rewriting tricks, and all of these mechanisms have different characteristics as well as different advantages and disadvantages.

In the specific case when the desire is to clone all messages sent to a particular channel to another channel while preserving the initial address that expanded to that channel, none of the previously mentioned methods provide that specific result. (A [capture action](#) in a [destination channel Sieve script](#) can capture the message, but not the initial address. [MESSAGE-SAVE-COPY](#) has similar limitations and also requires playing queue management games.)

The `clonehosts` channel option provides this result. It accepts a single argument: A space-separated list of host names. When given a `clonehosts` setting of "host1 host2 host3" and a message sent to the addresses `initial1` and `initial2`, both of which expanded to one or more final recipient addresses destined to that channel, this setting will add the recipient addresses:

```
@host1:initial1
@host2:initial1
@host1:initial2
@host2:initial2
```

33.3.12.3 Forced routing to gateways (`daemon`)

The interpretation and usage of the `daemon` channel option depends upon the type of channel to which it is applied. Currently, the only type of channel for which the `daemon` option is relevant is SMTP over [TCP/IP channels](#). Normally such channels connect to whatever host is listed in the envelope address of the message being processed. The `daemon` option is used to tell the channel to instead connect to a specific remote system, generally a firewall or mailhub system, regardless of the envelope address. The actual remote system name is given as an argument to `daemon`, *e.g.*:

```
msconfig> set channel:tcp_firewall.daemon firewall.domain.com
```

If the argument after the `daemon` option is not a fully qualified domain name (or alternatively a square bracket enclosed literal IP address), the argument will be ignored and the channel will connect to the channel's official host. When specifying the firewall or gateway system name as the channel's [official host name](#), `channel:channel-name.official_host_name`, the argument given to the `daemon` option is typically specified as `router`, *e.g.*:

```
msconfig> show channel:tcp_firewall
role.channel:tcp_firewall.official_host_name = firewall.domain.com
role.channel:tcp_firewall.daemon = router
role.channel:tcp_firewall.mx (novalue)
role.channel:tcp_firewall.pool = SMTP_POOL
```

```
role.channel:tcp_firewall.smtp (novalue)
```

33.3.12.4 Specify a last resort host for delivery (`lastresort`)

The `lastresort` channel option is used to specify a host to which to connect when all other connection attempts fail. In effect this acts as an MX record of last resort. This is only useful on [SMTP over TCP/IP channels](#).

Note that the `lastresort` host is attempted only for hosts that are in the DNS, having either MX records or an A record, and for whom the connection attempts to all the MX records -- or to the A record, if there were no MX records--have encountered temporary connection failures. (In particular, the `lastresort` host will not be attempted for a host that is only in the hosts file, not in the DNS at all. Also keep in mind that a permanent 5xx error in response to a connection attempt to a host is a permanent error, and will result in bouncing a message; in particular, the `lastresort` host will not be attempted after such a permanent rejection error. Also, the `lastresort` host will not be attempted if a connection succeeds, but the MTA's wait for an SMTP banner line to be returned times out; that again is not a temporary *connection* failure.)

This channel option requires a single parameter specifying the name of the "system of last resort".

See also the [IP_ACCESS mapping table](#), which can provide an alternate way of doing "fail over" for outbound IP connections for SMTP and LMTP channels.

Note that in most cases, it is preferable to fix problematic DNS records rather than to use `lastresort`; `lastresort` is intended merely for a few, special sorts of cases where correcting DNS records may not be possible, yet some "last ditch", MX-like, re-routing may be useful.

33.3.12.5 Multiple gateways on a single channel (`multigate`, `nomultigate`)

The `multigate` channel option tells the MTA to route the message to the daemon mailbox specified by the `daemon` channel option on the system specified in the message's To: address. This differs from the MTA's normal behavior when the `multigate` channel option is not used, in which case the MTA routes the message to the official host associated with the channel, *not* the system specified in the message's To: address.

There are a variety of caveats associated with using the `multigate` channel option; some of its former uses are now obsolete. The remaining usage is on [LMTP channels](#). `nomultigate` is the default.

33.3.12.6 `user` Option Under channel

The `user` channel option is used on [pipe channels](#) to indicate under what Unix user id to run.

In the 8.0 release and later, this option is deprecated and the `pipeuser` option from `restricted.cnf` is used instead.

33.3.13 Headers channel options

There are a number of channel options especially relevant regarding header line processing and handling; those channel options relating to header line processing *other* than address

handling in header lines are listed here. Note that channel options that relate more to general address handling (including address handling in headers) are instead listed primarily under [Addresses channel options](#).

33.3.13.1 Adding Return-path: header fields (`addreturnpath`, `noaddreturnpath`)

When specified on a destination channel, the `addreturnpath` channel option causes the MTA to add a Return-path: header field and possibly an Original-recipient: header field to all messages enqueued to the channel. `noaddreturnpath` disables this feature.

The Return-path: header field will contain the current envelope from (SMTP MAIL FROM) address enclosed in angle brackets. The Original-recipient: header field will contain the value of any SMTP ORCPT parameter associated with the message's recipient(s). Original-recipient: header fields are only generated when all recipients of the current copy of the message have the same ORCPT value.

Note that the addition of Return-path: and Original-recipient: header fields is usually a function of a final delivery agent (such as a final delivery channel). Most final delivery channels know to add these header fields themselves. But for the convenience of some channels such as the [ims-ms channel](#), where adding a Return-path: header line in the channel program itself is not convenient, the MTA is also capable of adding the Return-Path: header itself if configured to do so via this channel option.

The `addreturnpath` channel option is the default on the [ims-ms channel](#) and any channel marked with one of the `lmtplib*` channel options; `noaddreturnpath` is the default for all other channels.

33.3.13.2 Authenticated originator information processing (`authrewrite`)

The `authrewrite` option may be used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers. Normally the SMTP AUTH information is used (specifically, the user's canonical e-mail address, that is, the value of the `mail` attribute, found when looking up the user for authentication), though this may be overridden via the [FROM_ACCESS mapping](#). `authrewrite` takes a required bit-encoded integer value as an argument, according to the following table:

Table 33.8 `authrewrite` option values

Bit	Value	Usage
0-3	1	Add a Sender: header line, or a Resent-sender: header line if a Resent-from: or Resent-sender: was already present, containing the AUTH originator
0-3	2	Add a Sender: header line containing the AUTH originator
0-3	3	Use the AUTH_REWRITE mapping table , probing with any Resent-Sender: and Resent-From: info if present, and otherwise probing with Sender: and From: info
0-3	4	Use the AUTH_REWRITE mapping table , probing with Sender: and From: info
0-3	5	Add a From: header line, or a Resent-From: header line if a Resent-From: or Resent-Sender: was already present, containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!
0-3	6	Add a From: header line containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!

4	16	(New in 6.2) If set, apply the AUTH_REWRITE mapping table , even if SMTP AUTH has not been used
5	32	(New in 6.2) If set, probes to AUTH_REWRITE include the source-channel as a prefix field, separated by a vertical bar character from the rest of the probe string; that is, when this bit is set then probes take the form: <i>source-channel env-from [resent-]sender [resent-]from auth-originator</i>
6	64	(New in 7.2-7.02.) If set, use the rewritten version of the envelope from address in constructing the AUTH_REWRITE probe.
7	128	(New in 7.2-7.02.) If set, use the canonical version of the envelope from address in constructing the AUTH_REWRITE probe. Bit 6 (value 64) is a no-op if this bit is set.
8	256	(New in 7.3-11.01.) If set, add the value of the AUTH parameter from the SMTP MAIL FROM command to the AUTH_REWRITE probe, appearing just after the authorized originator address.

33.3.13.2.1 AUTH_REWRITE mapping table

Certain values of the [authrewrite channel option](#) cause the AUTH_REWRITE mapping table to be consulted to allow for more complex decision making and alterations of addresses. And bits of `authrewrite` also affect the form of probe to the AUTH_REWRITE mapping table.

Probes for the AUTH_REWRITE mapping table normally have the following format:

source-channel|env-from|[resent-]sender|[resent-]from|auth-originator|auth-parameter

Note that the `source-channel` field and its vertical bar suffix is only present if (new in JES MS 6.2) bit 5 (value 16) is set in the `authrewrite` argument, and `auth-parameter` and its vertical bar prefix are only present if (new in 7.3-11.01) bit 8 (value 256) is set in the `authrewrite` argument.

With `authrewrite 3`, the probes preferentially use any Resent-Sender: or Resent-From: header line values present, whereas with `authrewrite 4` the probes always use Sender: and From:. (Note that normally the AUTH_REWRITE mapping table is only consulted when a submission has included SMTP AUTH info; that is, in order for the AUTH_REWRITE mapping table to be consulted not only must the relevant incoming channel be marked with an `authrewrite` value of 3 or 4, but also the submission included use of the SMTP AUTH command. However, if bit 4 (value 16) is set in the `authrewrite` channel option's argument, then AUTH_REWRITE will be consulted even for non-authenticated submissions.)

New in 7.2-7.02, bit 6 (value 64) of `authrewrite` will, if set, cause a rewritten version of the envelope from address to be used for the `env-from` address in the probe as opposed to the original form given in the SMTP MAIL FROM command. The specific rewritten form used is controlled by bit 7 (value 128): If set the canonical form return address will be used, if clear the normally rewritten form will be used instead. These rewritten forms are useful when accessing checking is done using the AUTH_REWRITE mapping in order to prevent envelope from forgery by authenticated users.

New in 7.0.5, if bit 9 (value 512) of `authrewrite` is set, the final tag set by the `*_ACCESS` mappings will be prefixed to the AUTH_REWRITE mapping probe.

As of the 8.0 release, the following input flags will be set:

- \$A if SASL authentication has succeeded
- \$E if EHLO (EMSMTP) was used
- \$L if LHLO (LMTP) was used
- \$P if POP-before-SMTP was used

- \$R if this is an internal channel enqueue operation, *i.e.*, from a [conversion](#), [process](#), [reprocess](#), or similar sort of channel
- \$T if a SSL/TLS security layer has been negotiated

If the mapping table output contains a \$J, \$j, \$K, or \$k, then the envelope From address is replaced with the specified string. If the mapping table output contains a \$Y, \$y, \$T, or \$t, then a Sender: header line is added (if `authrewrite 3` was specified and if a Resent-Sender: or Resent-From: was already present, then a Resent-Sender: header line is added instead of a Sender: header line) containing the specified string.

If the mapping table output contains a \$Z or \$z, then a From: header line is added (a Resent-From: in the case of `authrewrite 3` and a Resent-From: or Resent-Sender: header line already being present) containing the specified string. (Such replacing of the From: header address is **NOT RECOMMENDED** and **CONTRARY TO INTERNET STANDARDS** and quite likely to **HARM** the overall security of your users. It should almost **NEVER** be done: **THIS MEANS YOU!** Despite the wishes and mistaken notions of many sites and users, the From: header line, in Internet e-mail, is **NOT INTENDED** to represent the "real" originator of a message; it is intentionally defined permitting alternate usages.)

New in 7.3-11.01, if a \$O is specified, then another vertical-bar-separated string will be read from the mapping result string and used to set or override the value of the SMTP AUTH parameter for the current transaction. The `saslpassth` channel option may then be applied to the destination channel to cause this value to be propagated as an AUTH parameter on the SMTP MAIL FROM command.

New in JES MS 6.2, if a \$N is specified, then the message will be rejected. Optional rejection text may be specified after another vertical bar character, |. And as of JES MS 6.3, \$X may also be used to specify the extended error code (specified before the \$N text, separated by a |) in the form `x.y.z`. In the absence of such optional text and optional extended error code, the default text "invalid originator address used" and default extended error code 5.7.0 will be used.

When using multiple such flags, separate the string arguments with the vertical bar character, |, and specify the string arguments in the order listed in the paragraph above; that is,

```
$J$Y$Z#env-from|sender|from-header
```

or

```
$X$N|error-code|rejection-text-string
```

Technically, one could use all five flags in the same entry, though it does not seem likely to be useful:

```
$J$Y$Z$X$N#env-from|sender|from-header|error-code|rejection-text-string
```

33.3.13.3 Comments in address message headers (`commentinc`, `commentmap`, `commentomit`, `commentstrip`, `commenttotal`, `sourcecommentinc`, `sourcecommentmap`, `sourcecommentomit`, `sourcecommentstrip`, `sourcecommenttotal`)

The MTA only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process comments (strings enclosed in parentheses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

On destination channels, this behavior is controlled by the use of the `commentinc`, `commentmap`, `commentomit`, `commentstrip`, and `commenttotal` channel options. `commentinc` tells the MTA to retain comments in header lines. It is the default. `commentmap` tells the MTA to apply the [COMMENT_STRINGS mapping table](#) to comments in addressing header lines if such a mapping table exists, while if no such mapping table exists then `commentmap` is equivalent to `commentstrip`. `commentomit` tells the MTA to remove any comments from addressing headers, *e.g.*, To:, From:, Cc: headers, *etc.* `commenttotal` tells the MTA to remove any comments from all headers, except Received: headers; as such, this option is not normally useful or recommended. And finally, `commentstrip` tells the MTA to strip any nonatomic characters from all comment fields.

On source channels, this behavior is controlled by the use of the `sourcecommentinc`, `sourcecommentmap`, `sourcecommentomit`, `sourcecommentstrip`, and `sourcecommenttotal` channel options. `sourcecommentinc` tells the MTA to retain comments in header lines. It is the default. `sourcecommentmap` tells the MTA to apply the [COMMENT_STRINGS mapping table](#) to comments in incoming addressing header lines if such a mapping table exists, while if no such mapping table exists then `sourcecommentmap` is equivalent to `sourcecommentstrip`. `sourcecommentomit` tells the MTA to remove any comments from addressing headers, *e.g.*, To:, From:, Cc: headers, *etc.* `sourcecommenttotal` tells the MTA to remove any comments from all headers, except Received: headers; as such, this option is not normally useful or recommended. And finally, `sourcecommentstrip` tells the MTA to strip any nonatomic characters from all comment fields.

These options can be applied to any channel.

33.3.13.4 Two or four digit date conversion (`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! (Please note that systems which cannot handle both formats are in violation of the standards.)

The `datefour` and `datetwo` channel options control the MTA's processing of the year field in message header dates. `datefour`, the default, instructs the MTA to expand all year fields to four digits. Two digit dates with a value less than 50 will have 2000 added while values greater than 50 will have 1900 added.

`datetwo` instructs the MTA to remove the leading two digits from four digit dates. This is intended to provide compatibility with noncompliant mail systems that require two digit dates; it should never be used for any other purpose.

33.3.13.5 Day of week in date specifications (`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` channel options control the MTA's processing of day of the week information. `dayofweek`, the default, instructs the MTA to retain any day of the week information and to add this information to date/time headers if it is missing.

`nodayofweek` instructs the MTA to remove any leading day of the week information from date/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.

33.3.13.6 Host name to use when correcting incomplete addresses (`remotehost`, `noremotehost`, `defaulthost`, `nodefaulthost`)

The MTA often receives from misconfigured or incompliant mailers and SMTP clients addresses which do not contain a domain name. The MTA, showing at least some respect for standards, must attempt to make such addresses legal before passing the message along. The MTA does this by appending a domain name to the address (*e.g.*, appends "@acme.com" to "mrochek"). For envelope To addresses missing a domain name, the MTA normally assumes that the local host name should be appended -- however, see the detailed discussion of the `defaulthost` option below for some exception cases. (Exceptions: unless the `defaulthost` option is used and either the channel is a `submit` channel so that the `defaulthost` option's first argument applies also to envelope To addresses, or a second parameter (that has at least one period in it) is specified for the `defaulthost` option.) However for other addresses, such as From: addresses, in the case of the MTA's SMTP server there are at least two reasonable choices for the domain name: the local MTA host name, or 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, for email from arbitrary remote sources, 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 mail client such as a POP or IMAP client that uses SMTP to post messages. Or if lightweight mail clients such as POP or IMAP clients "ought" to have their own specific domain name which is not that of the local host, then adding 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 `remotehost` channel option specifies that the remote host's name should be used. The `noremotehost` channel option specifies that the local host's name should be used and that any existing `remotehost` setting be reset. The default to neither use the remote host's name nor reset any existing setting.

The `defaulthost` channel option is used to specify a particular host name to append to incoming bare usernames; it must be followed by the domain name to use in completing addresses (in envelope From: and in headers) coming in that channel. An optional second domain name (that contains at least one period) may be specified to use in completing envelope To addresses. `nodefaulthost` is the default. `defaulthost` overrides `remotehost` if both are set.

Note that the `switchchannel`, `saslswitchchannel`, `tlsswitchchannel` and various other options can be used to associated incoming SMTP connections with a particular channel other than the one the SMTP server uses by default. The default and remote host settings on the final channel that's switched to will be the ones that are used. Note, however, that once `remotehost` has been activated only an explicit `noremotehost` setting will deactivate it.

This facility can be used to group remote mail clients on a channel where they will receive proper treatment. Alternatively, it is an unconditionally simpler proposition to deploy standards-compliant remote mail clients (even if a multitude of incompliant clients are in use) rather than attempting to fix the network-wide problem on your MTA hosts.

33.3.13.7 Strip illegal blank recipient headers (`dropblank`, `nodropblank`)

In [RFC 822](#) 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 option, if specified on a source channel, causes the MTA to strip any such illegal blank headers from incoming messages. `nodropblank` disables this action and is the default.

33.3.13.8 Envelope tunneling via header fields (`forcedreceivedfrom`)

The `envelopetunnel` channel option controls the transfer of envelope information to and from special header fields defined for this purpose. The use of these fields provides a means of tunneling information associated with various SMTP extensions through systems that do not support the extensions. The option's value is a bit-encoded integer, with each bit controlling a different header field and associated piece of envelope information. Setting the bit enables tunneling; clearing it disables it.

The default value of this option is 0, meaning all tunneling is disabled. Setting the option to -1 enables all available tunneling capabilities.

At present only one bit is defined: Bit 0 (value 1) controls the use of the MT-Priority: header line as a means of tunneling the message's MT-PRIORITY value. The syntax and use of the field are specified in [RFC 6758](#).

33.3.13.9 Header-based message expiration(`expirysource`, `expirysource`)

(New in Messaging Server 7.0.) The `expirysource` channel option instructs the MTA to honor Expiry-date: header fields - messages will be returned as undeliverable if the time specified by this header field is exceeded. `noexpirysource` disables this check and is the default.

33.3.13.10 Syntax Error Fixup (`fixsyntaxerrors`, `passyntaxerrors`)

The MTA normally attempts to fix common syntax errors when it processes message header fields. It is sometimes useful to disable this behavior so certain syntax errors aren't corrected. The `passyntaxerrors` source channel keyword enables this behavior. `fixsyntaxerrors` is the default.

At present disabling syntax error fixup is limited to header fields containing addresses and message ids. This may be extended to include other types of fields in the future.

Important note: Many other agents depend on syntax error fixup by the MTA. Disabling it may fix some problem but cause unexpected side effects. It is always preferable to fix whatever is generating the syntax errors so it doesn't do that any more.

33.3.13.11 Location of message header (`headerbottom`, `headerinc`, `headeromit`)

VMS MAIL only provides support for four message header lines: From:, To:, Cc: and Subject:. However, [RFC 822](#) headers can contain many additional types of header lines. On OpenVMS systems, PMDF supports these additional header lines by optionally prepending or appending them to the message body whenever a message is delivered to a local user.

This behavior is controlled by the use of the `headerinc`, `headeromit`, and `headerbottom` channel options. The default is `headerinc`, which tells PMDF to prepend the header lines to the message. On OpenVMS systems, the channel option `headerbottom`, which tells PMDF to append the header lines to the end of the message, and `headeromit`, which tells PMDF to strip all header lines, are also available.

In some rare cases an SMTP server is used to deliver message content to something other than a proper mail user agent. Under these very rare circumstances it may be appropriate to omit header information or to place it at the end of the message, so these channel options are also supported for SMTP channels.

Extreme care should be taken not to use these options on channels connecting to other message handling systems --- relocating or eliminating message headers violates [RFC 821](#) and [RFC 822](#) and can lead to serious problems. Note also that many seemingly "bothersome" header lines may contain valuable information required to decode messages, track down problems, or even authenticate who really sent the message and thus thwart attempts at forging mail messages.

33.3.13.12 Cutting header to fit (`headercut`)

The `headercut` channel option cuts the current message header down to no more than the specified number of bytes using a heuristic algorithm that removes or truncates header fields based on their relative importance. Unlike `headertrim`, which should be used to deal with issues of the mere presence, or the number, or the size, of specific header fields, `headercut` is intended for use to meet constraints on overall header size.

The `headercut` option requires a single nonnegative integer argument. A value of 0, the default, disables header cutting.

33.3.13.13 Header alignment and folding (`headerfoldpreserve`, `headerfoldremove`, `headerlabelalignment`, `headerlineincrement`, `headerlinelength`)

The `headerlabelalignment` channel option 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 headers with an alignment point of 10 would appear as follows:

```
To:      ned@innosoft.com
From:    kristin@innosoft.com
Subject: Alignment test
```

The default `headerlabelalignment` is 0, which causes headers not to be aligned.

The `headerlinelength` channel option controls the length of message header lines enqueued on this channel. The default, if this channel option is not explicitly set, is 80.

Lines longer than this are folded in accordance with [RFC 822](#) folding rules. Note that `headerlinelength` applies to all header lines; when some header lines should get different folding points than other header lines, then see instead the `LINELENGTH` [header trimming option](#).

The MTA attempts to fold the lines at or before the length specified by `headerlinelength`, but if no suitable folding point can be found, then the length is adjusted by `headerlineincrement`. `headerlineincrement` takes a required non-negative integer argument. Only values between (inclusive) 1 and 100 are permitted; note that setting values that diverge dramatically from the default value of 20 may result in problematic behavior.

Note that these channel options 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 transported across the Internet, so these channel options may have no visible effect even when used in conjunction with simple user agents that do not reformat message headers.

Whether the MTA attempts to preserve "original" fold points in folded header lines when practicable, or whether the MTA automatically unfolds and then later re-folds header lines, is controlled by the `headerfoldremove` and `headerfoldpreserve` channel options. These options apply to source channels.

`headerfoldremove` is the default; it means that the MTA unfolds all incoming header lines when first receiving a message (and then refolds when outputting header lines, at fold points chosen in accordance with the `headerlinelength` channel option or `LINELENGTH` [header trimming option](#)).

The `headerfoldpreserve` channel option tells the MTA to attempt to preserve "original" fold points in header lines. Note that fold points in addressing header lines (such as `To:`, `Cc:`, `Bcc:`, *etc.*) are not preserved by this channel option, nor are fold points in date header lines or MIME header lines. This keyword instead affects primarily text header lines such as the `Subject:` header line. When this option is used, it is usually most appropriate to also set [headertrailingpreserve](#); that is, when attempting to preserve original fold points, typically one also wants to preserve all original white space, including trailing white space, that might be present in the header line. However, note that enabling the `headerfoldpreserve` channel option causes any originally present horizontal tab characters to be converted to space characters, except that immediately after each fold point a horizontal tab character will be used as the initial linear white space character (regardless of whether the original white space character was a tab or a space).

Note that various aspects of this sort of header line processing can be testing using the [test - header utility](#).

Note that message bodies are potentially subject to MIME encoding to ensure transport-safe line length, as controlled by the `linelength` channel option; such message body encoding is separate and distinct from (and happens after) the header line processing discussed here.

33.3.13.14 Trimming message header lines (`headertrim`, `noheadertrim`, `headerkeeporder`, `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 option and an associated header option file or two. The `headertrim` channel option 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` option bypasses header trimming, but does cause the MTA to re-order header lines according to its internal header line order defaults (plus any [configured ordering rules](#)). New in JES MS 6.3 is the `headerkeeporder` channel option, which (making use of new MTA header line handling) preserves whatever header line ordering was present in the message as enqueued. `noheadertrim` was the default in JES MS 6.2 and earlier; as of JES MS 6.3, `headerkeeporder` is the default.

The `innertrim` channel option instructs the MTA to perform header trimming on inner message parts, *i.e.*, embedded MESSAGE/RFC822 parts, as well. Note that setting `innertrim` overrides `headerkeeporder`, causing at a minimum `noheadertrim` effect (that is, potential re-ordering of header lines). The `noinnertrim` channel option, which is the default, tells the MTA not to perform any header trimming on inner message parts.

The `headerread` channel option 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 destination channel, rather than source channel, related. The `noheaderread` channel option bypasses message enqueue header trimming. `noheaderread` is the default.

Unlike the [headeromit](#) and [headerbottom](#) options, the `headertrim` and `headerread` options may be applied to any channel whatsoever. Note, however, that stripping away vital header information from messages may cause improper MTA operation. 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. *Do not merely trim header lines away because you or your users find them annoying --- those header lines are there for a reason. More often than not, the header lines that users feel are superfluous are among the most important. Before trimming or removing any header line, be sure that you understand the usage of that header line and have considered the possible implications of its removal.*

Header options files for the `headertrim` and `innertrim` channel options 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 `headerread` channel option have names of the form `channel_read_headers.opt`. See [Header option files](#) for information on the format of these files.

Note that as of JES MS 6.3, the MTA supports the [Sieveeditheader](#) extension, which offers quite a different way to modify message header lines.

33.3.13.15 Limiting header storage (`headerlimit`)

The MTA stores outermost message headers in memory. This means that a single extremely large header could potentially consume all available memory, possibly leading to a denial of service attack. When placed on a source channel, the `headerlimit` channel option provides a means to prevent such attacks. It accepts a single integer argument specifying the maximum number of MTA blocks that can be consumed by the message header. Messages with headers that exceed this limit will be silently truncated.

Note that the `header_limit` MTA option can be used to implement the same limit for all channels. The lower of the two option values is used as the actual limit.

The default value for `headerlimit` is essentially unlimited if the channel option isn't specified. Note, however, that since the default value for the `header_limit` MTA option

is 2000 blocks, the effective default limit on the size of a header if no limiting options are specified is 2000 blocks.

The `headerlimit` channel option initially appeared in Messaging Server 6.1.

33.3.13.16 Trailing spaces on header lines (`headertrailingpreserve`, `headertrailingremove`)

The `headertrailingremove` and `headertrailingpreserve` channel options, applicable on source channels, control whether the MTA strips (removes) trailing white space from the ends of physical header lines.

By default (`headertrailingremove`) the MTA removes trailing white space from the ends of physical header lines; this means both from the ends of logical (unfolded) header lines, and from the ends of folded portions (physical lines) of a single (logical) header line.

The `headertrailingpreserve` channel option may be used to instead preserve such trailing white space. This may be of particular interest for folded header lines, when it is desired to preserve original, "interior" white space in the logical (unfolded) header line. It is also typically used in conjunction with the `headerfoldpreserve` channel option.

33.3.13.17 Inner header rewriting (`inner`, `noinner`)

The MTA only interprets the contents of header lines 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` channel options on destination channels. `noinner` tells the MTA there is no special need to rewrite inner message header lines (though inner message header line processing may be triggered by other MTA facilities). It is the default. `inner`, when placed on a destination channel, tells the MTA to parse messages and rewrite inner headers.

These options can be applied to any channel.

33.3.13.18 Default language tag (`language`)

Certain MTA operations, especially those involving selection of textual content to send to users, may need to take language into account. For example, nondelivery notification text may be available in multiple languages and the `NOTIFICATION_LANGUAGE mapping` and `DISPOSITION_LANGUAGE mapping` can be used to select the appropriate language to use.

The MTA also supports use (and selection amongst) language-tagged values for various `mailAutoReply*` LDAP attributes used to construct vacation messages; see the discussion of the `ldap_autoreply_subject` and `ldap_autoreply_text*` MTA options in particular.

It is axiomatic that in order to make decisions based on language language tagging information must be available. Normally this information is derived from the message being processed, *e.g.*, from an `Accept-language:` header field, from a `Preferred-language:` header field, or even from the country code found in the `From:` address. However, not all messages contain such information.

The `language` channel option can be used to associaed a default language with a particular source channel. A single string argument is required specifying a language tag. Messages

originating from this channel which aren't tagged in any other way will be effectively tagged with this value. The default is not to assume any language tag on a per-channel basis.

33.3.13.19 Detecting the end of the message header (`limitheadertermination`, `relaxheadertermination`)

Message headers consist of a series of fields, each folded onto one or more lines. As a result a header consists of, at the outermost syntactic level, CRLF terminated lines that begin with either one or more whitespace characters or an alphanumeric label followed by a colon. In all cases the header is supposed to be terminated by a CRLF CRLF sequence.

A header line that doesn't meet these syntactic requirement can arise in two different ways: (1) Something emitted a syntactically invalid header line or (2) The CRLF CRLF separator is missing and message content has been elided with the preceding header.

By default, the MTA handles syntactically invalid lines as part of the message content, terminating header processing. Setting the `limitheadertermination` channel option on an incoming (source) channel) will cause the MTA to treat such lines as part of the header and continue header processing.

[RFC 822](#) was ambiguous as to whether a line containing merely white space was allowed in a message header. ([RFC 5322](#) clarifies this by disallowing the generation of such a line in a message header and requiring that it be accepted as part of a header.) By [RFC 822](#) rules, it is ambiguous whether such a line should be interpreted as the end of the message header (interpreted as the "blank" line separating message header from message body), or whether it is merely additional white space from the previous header line "folded" onto a new line.

By default, the MTA only interprets a strict CRLF CRLF sequence as the end of the message header. Setting the `relaxheadertermination` channel option on an incoming (source) channel) will cause the MTA to also interpret lines that contain merely white space (spaces or TABs) as terminating the message header.

`limitheadertermination` is the default, though it did not exist as a distinct channel option prior to MS 7.0.5. (In earlier versions, this default behavior was selected by *not* setting the `relaxheadertermination` channel option.)

33.3.13.20 Automatic splitting of long header lines (`maxheaderaddr`s, `maxheaderchar`s)

Some message transports, notably some older 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 fairly common problem.

The MTA provides per-channel facilities to split (break) long header lines into multiple, independent header lines. The `maxheaderaddr`s channel option controls how many addresses can appear on a single line. The `maxheaderchar`s channel option controls how many characters can appear on a single line. Both channel options require a single integer argument that specifies the associated limit. By default, no limit is imposed on the length of a header line nor on the number of addresses which may appear.

33.3.13.21 Legalizing messages that lack any recipient headers (`missingrecipientpolicy`)

[RFC 822](#) (Internet) messages are required to contain a recipient header line: a To:, Cc:, or Bcc: header line. A message without any such header line is illegal according to [RFC 822](#). Nevertheless, some broken user agents and mailers (e.g., many older versions of sendmail) will emit such illegal (per [RFC 822](#)) messages.

Note that [RFC 2822](#), the update to [RFC 822](#), relaxes the [RFC 822](#) requirement and allows submitted messages to be lacking in any recipient header line. However, unless it is certain that *all* the MTAs and MUAs that may ever handle a message in fact conform to [RFC 2822](#) (rather than the older [RFC 822](#)), it is unwise to emit a message lacking all recipient header lines, since the behavior of an [RFC 822](#)-compliant MTA or mail user agent may be undesirable when encountering a message that is, from its point of view, illegal--results may include rejection of such a message, potentially undesired exposure of recipient information such as recipients intended as Bcc: recipients, *etc.*

The `missingrecipientpolicy` channel option takes an integer value specifying what approach to use for such messages; the default value, if the channel option is not explicitly present, is to use the MTA option `missing_recipient_policy` value (which itself defaults to 0, if not set, which as of JES MS 6.2 is equivalent to a value of 1 meaning that messages are passed through unchanged--in JES MS 6.0 and JES MS 6.1 the default value of 0 had been equivalent to a value of 2 meaning that envelope To: addresses are placed in a To: header).

Table 33.9 missingrecipientpolicy MTA option values

Value	Action
0	Use current best practices to resolve the situation. Prior to 6.2 this was the same as 2, in 6.2 and later it is the same as 1.
1	Pass the illegal-per- RFC 822 (though legal per RFC 2822) message through unchanged.
2	Place envelope To: recipients in a To: header.
3	Place all envelope To: recipients in a single Bcc: header.
4	Generate an empty group construct (<i>i.e.</i> , ;) To: header line. The phrase used in the group construct is controlled by the <code>missing_recipient_group_text</code> MTA option, so for instance "To: Recipients not specified: ;".
5	Generate a blank Bcc: header.
6	Reject the message (with a "554 5.6.0 Error writing message - message is missing required recipient header fields" error).

Note that the `missing_recipient_policy` MTA option can be used to set an MTA system default for this sort of behavior.

33.3.13.22 Envelope to address in Received: header (`receivedfor`, `noreceivedfor`, `receivedfrom`, `noreceivedfrom`)

The `receivedfor` channel option instructs the MTA that if a message is addressed to just one envelope recipient, to include that envelope To: address in the Received: header it constructs. In such cases, the envelope To: address will be noted within the Received: header line via a clause of the form:

for *recipient*

The `receivedfor` channel option is the default. The `noreceivedfor` channel option instructs the MTA to construct Received: headers without including any envelope addressee information.

The `receivedfrom` channel option instructs the MTA to include the original envelope From: address when constructing a Received: header for an incoming message if the MTA has changed the envelope From: address due to, for instance, certain sorts of mailing list expansions; in such cases the original envelope From: address will be noted in the Received: header line via comment of the form

(original mail from *original-envelope-from*)

The `receivedfrom` channel option is the default. The `noreceivedfrom` channel option instructs the MTA to construct Received: headers without including the original envelope From: address.

See also the [\[RECEIVEDFOR\]](#), [\[NORECEIVEDFOR\]](#), [\[RECEIVEDFROM\]](#), and [\[NORECEIVEDFROM\]](#) alias file named parameters, or in Unified Configuration, the alias options `alias_receivedfor`, `alias_noreceivedfor`, `alias_receivedfrom`, and `alias_noreceivedfrom`, which all override the channel settings on a per-alias (or per-mailing list) basis.

33.3.13.23 Generation of X-Envelope-to: header lines (`x_env_to`, `nox_env_to`)

The `x_env_to` and `nox_env_to` channel options 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` channel option, the `x_env_to` channel option enables generation of these headers while the `nox_env_to` will remove such headers from enqueued messages. The default is `nox_env_to`; (note that this default behavior is a change in PMDF V5.0 from previous versions of PMDF). Note that the fact that `x_env_to` also requires the `single` channel option in order to take effect represents a change of behavior from PMDF V5.1 and earlier.

The (non-standard) X-Envelope-to: header line contains a duplicate of the recipient addresses that appear in the message envelope. The X-Envelope-to: header line for a particular copy of a message contains only the addresses that that particular copy is being sent to; it does *not* contain all the addresses on the header To: line (except in the simplest case where only a single copy of the message is needed). Also note that there can be envelope recipient addresses that are mentioned nowhere in the header due to the use of things like autoforwarders or blind carbon copies.

New in 6.3, the `x_env_to` channel option no longer requires accompanying use of the `single` channel option in order to take effect. If used without `single`, the header will simply contain a comma-separated list of all recipients this copy of the message is intended for. This can be useful in situations where the current recipient list needs to be provided as part of the header to some processing agent.

Note that the full recipient list for a message can, if retained at the time of final delivery, reveal the presence of blind carbon recipient to the other, regular recipients. As such, this facility should only be used when it is absolutely necessary.

33.3.13.24 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA provides support for Postfix's XCLIENT SMTP extension. The PostFix documentation for the extension can be found here:

http://www.postfix.org/XCLIENT_README.html

Use of XCLIENT is controlled by three main source channel keywords, `noxclient`, `xclient`, and `xclientsasl`, and variants `xclientrepeat` and `xclientsaslrepeat`. `noxclient` is the default, and means that XCLIENT is not advertised in the response to EHLO and the XCLIENT command itself is disabled. If `xclient` is set the XCLIENT command is enabled and the NAME, ADDR, PORT, PROTO, and HELO attributes may be used. `xclientsasl` enables the LOGIN attribute in addition to all the others. It should be noted that LOGIN specifies an external identity that must then be bound to the session identity through the use of SASL EXTERNAL.

By default, only one set of XCLIENT commands is allowed in a single SMTP session. Specifying `xclientrepeat` allows groups of XCLIENT commands to be repeated, allowing a proxy or similar agent to share a connection between multiple clients. `xclientsaslrepeat` allows multiple groups of XCLIENT commands including LOGIN. Note that care should be taken when these keywords are used since the server cannot determine the origin of a given XCLIENT command.

The primary visible effect of XCLIENT is on the contents of the Received: field the MTA adds. For example, if this XCLIENT command was executed:

```
xclient name=foo.domain.com addr=1.2.3.4 helo=bar.domain.com port=12345
```

it would result in a header of the general form:

```
Received: from bar.domain.com (foo.domain.com [1.2.3.4])
  by server.domain.com (Oracle Communications Messaging Server 7.0.5.32
  64bit (built Aug 18 2014)) with imapsubmit
  id <01OJ9P51WPFC007KNZ@server.domain.com> for user@domain.com;
  Mon, 20 Aug 2012 08:17:31 -0700 (PDT)
```

However, the ADDR and PORT attributes also change the contents of the transportinfo that appears in various mapping table probes, such as the probe to `PORT_ACCESS`. Given the preceding XCLIENT command, the transportinfo part of the mapping probes would change to something like:

```
TCP|<destaddr>|25|1.2.3.4|12345
```

33.3.13.25 Personal names in address message headers (`personalinc`, `personalmap`, `personalomit`, `personalstrip`, `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, `sourcepersonalstrip`)

The MTA only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process personal names (strings preceding angle-bracket-delimited addresses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

On destination channels, this behavior is controlled by the use of the `personalinc`, `personalmap`, `personalomit`, and `personalstrip` channel options. `personalinc` tells the MTA to retain personal names in the headers. It is the default. `personalmap` tells the MTA to apply the `PERSONAL_NAMES` mapping table to personal names appearing in addressing header lines if such a mapping table exists, while if no such mapping table exists then `personalmap` is equivalent to `personalstrip`. `personalomit` tells the MTA to remove all personal names. And finally, `personalstrip` tells the MTA to strip any nonatomic characters from all personal name fields.

On source channels, this behavior is controlled by the use of a `sourcepersonalinc`, `sourcepersonalmap`, `sourcepersonalomit`, or `sourcepersonalstrip` channel option. `sourcepersonalinc` tells the MTA to retain personal names in the headers. It is the default. `sourcepersonalmap` tells the MTA to apply the `PERSONAL_NAMES` mapping table to personal names appearing in addressing header lines if such a mapping table exists, while if no such mapping table exists then `sourcepersonalmap` is equivalent to `sourcepersonalstrip`. `sourcepersonalomit` tells the MTA to remove all personal names. And finally, `sourcepersonalstrip` tells the MTA to strip any nonatomic characters from all personal name fields.

These options can be applied to any channel.

33.3.13.26 State clause in Received: header field (`receivedstate`)

Received: header fields can contain a "state" clause which indicates the type of processing a message is about to undergo; such state clauses were defined in [RFC 6729 \(Indicating Email Handling States in Trace Fields\)](#). The primary use for these clauses is to indicate when an operation is being undertaken that could cause a delivery delay, *e.g.*, the message has been quarantined.

The `receivedstate` destination channel option controls the generation of state clauses. A single argument is required which specifies the state name to insert into the Received: fields of messages enqueued to the corresponding channel. The default, when `receivedstate` is not specified, is not to insert any state clause.

State values are restricted by [RFC 6729](#) to be a state keyword "token" (as defined in [RFC 2045](#)), several such keywords being already registered, optionally followed by a slash character and another (unregistered) token as a detail label. Any token must consist solely of US-ASCII characters not including space, control characters, or the so-called "tspecials" characters:

```
tspecials := "(" / ")" / "<" / ">" / "@" /
            "," / ";" / ":" / "\" / <">
            "/" / "[" / "]" / "?" / "="
```

The initial set of registered state keywords is: `auth`, `content`, `convert`, `moderation`, `normal`, `other`, `outbound`, `quarantine`, and `timed`.

Appropriate usage with the MTA could include:

```

msconfig> show channel:*.receivedstate
role.channel:conversion = convert
role.channel:defragment = convert/defragment
role.channel:filter_discard = quarantine/sieve-discarded
role.channel:reprocess = other
role.channel:tcp_intranet = normal
role.channel:tcp_local = outbound
    
```

with use on the [conversion channel](#), [defragment channel](#), [filter_discard channel](#), and [reprocess channel](#) being particularly relevant to note message transitions/points where messages could be delayed.

33.3.13.27 Mapping Reply-to: header when gatewaying to non RFC 822 environments (usereplyto)

The `usereplyto` channel option controls the mapping of the Reply-to: header in certain non [RFC 822](#) environments. (Currently, the `usereplyto` channel option is relevant only for the OpenVMS local channel, and the PMDF-LAN cc:Mail, GroupWise, and Microsoft® Mail channels.) The default argument for `usereplyto` is 0, which means to use the channel default behavior (which varies from channel to channel).

Table 33.10 `usereplyto` MTA option values

Value	Action
-1	Never map Reply-to: addresses to anything.
0	Use the channel default mapping of Reply-to: addresses; (varies from channel to channel). This is the default.
1	Map Reply-to: to From: if no usable From: address exists.
2	If there is a usable Reply-to: address, then map it to From:; otherwise fall back to the From: address.

33.3.13.28 Mapping Resent- headers when gatewaying to non RFC 822 environments (useresent)

The `useresent` channel option controls the use of Resent- headers when gatewaying to environments that do not support [RFC 822](#) headers. This channel option takes a single integer-valued argument. Legal values include:

Table 33.11 `useresent` MTA option values

Value	Action
+2	Use any Resent- headers that are present to generate address information.
+1	Only use Resent-From: headers to generate address information; all other Resent- headers are ignored.
0	Do not use Resent- headers to generate address information. This is the default.

Currently the `useresent` channel option applies for the l (lowercase "L") channel on OpenVMS, and for PMDF-MR, PMDF-X400, and some PMDF-LAN channels.

Note that the default of 0 constitutes a change in the behavior of the OpenVMS 1 channel compared with PMDF version 4.3 and earlier.

33.3.14 Host name channel options

There are three options that, for any type of channel, configure the fundamentals of a channel's own "host name" (`official_host_name`), the channel's knowledge of the host name of the system on which it is operating (`local_host_alias`), and optionally remote "hosts" with which it communicates (`additional_host_names`); these three basic options correspond to separate syntactic components of channel configuration in legacy configuration, but are set via options in Unified Configuration.

In addition to these three fundamental options, other channel options relating (in the case of TCP/IP channels) both to a system's own host name and to remote host names include `daemon`, `lastresort`, and the set of `defaulthost`, `nodefaulthost`, `remotehost`, and `noremotehost` channel options.

33.3.14.1 `official_host_name` Option

The `official_host_name` channel option specifies the "name" of the system with which this channel communicates. The `official_host_name` may be either a fully qualified host name, in the case of a channel dedicated to communicating with one particular system,

```
msconfig> show channel:l.official_host_name
role.channel:l.official_host_name = host.domain.com
```

or in the case of a more "generic" channel used for communicating with multiple systems (such as the Internet-communication channel `tcp_local`), the `official_host_name` tends to be a generic, place-holder name, *e.g.*,

```
msconfig> show channel:tcp_local.official_host_name
role.channel:tcp_local.official_host_name = tcp-daemon
```

In legacy configuration, the official host name is specified as the first name on the second line of a channel definition:

```
tcp_local ...keywords...
tcp-daemon
```

Each channel must have its own, unique `official_host_name`; no duplication with other channels is allowed. As of JES MS 6.1, the official host name is limited to 128 characters; in prior versions the limit was 40 characters. An official host name is required for each channel; omitting an official host name from a channel definition is an error.

Note that the `official_host_name` on the `l` channel (lowercase "L" channel) has somewhat special significance, as it is used/assumed at certain times by the MTA; and normally, it would be set to match the value of the `ldap_local_host` MTA option.

As of 8.0.1, the option if not set for a channel will default to `channel-name.hostname` in unified configurations.

33.3.14.2 `local_host_alias` Option

The `local_host_alias` channel option is used to specify an alternate name for the MTA system itself. Normally, the local host system is known by the name that appears as the `official_host_name` on the `l` channel. But it is sometimes useful for the local host to have different names depending on the channel being used. This situation usually arises when a machine is connected to more than one network. For example, a system may need to be known as `ymir.uucp` on the UUCP network, `ymir.claremont.edu` on the Internet, and `ymir.bitnet` on BITNET.

If `local_host_alias` is specified, it is communicated as the local host's name to any remote hosts with which this channel communicates. This alias will replace the local host's name wherever it appears in the envelope and header of messages queued to the associated channel. If this alias is omitted the local host's official name (that is, the official host name associated with the `l` channel, `channel.l.official_host_name`) is used.

In legacy configuration, a local host alias is specified by placing it on the second line of a channel definition, subsequent to the official host name:

```
channel-name ...keywords...
official-host-name local-host-alias
```

The local host alias only affects the name of the local host. No other system names are affected. The effects of the local host alias are strictly limited to the channel to which the alias applies.

A `local_host_alias` on the SMTP server channel (typically `tcp_local`) overrides the TCP/IP stack's official host name for use on the SMTP server's 220 banner line. (If the TCP/IP stack doesn't have a value for some reason, we finally fall through to the `official_host_name` from the `l` channel.) A `local_host_alias` on an outgoing `tcp_*` channel overrides the TCP/IP stack's official host name for use on the EHLO/HELO/LHLO line. As of iMS 5.2, the new-in-that-version `BANNER_HOST` TCP/IP-channel-specific option takes precedence over the `local_host_alias` (in both directions, server and client).

Note: The use of local host aliases is discouraged. If at all possible, each system should be known by one and only one name on all networks. Networks should strive to make this a reality. (Back when Internet *vs.* UUCP *vs.* BITNET names were an issue, when it was impossible for a host to have the same name on both BITNET and the Internet, local host aliases were a necessary feature. Since different networks are always associated with different channels, a per-channel local host alias was an ideal way to give the local host a per-network name.) Note especially that when a single network is involved, it may appear that local host aliases can solve lots of problems, but often the end result is a worse mess than if the proper course of action is selected --- pick a single name and stick to it, living with the consequences of the conversion now instead of putting them off until it becomes even more difficult.

33.3.14.3 additional_host_names Option

In Unified Configuration, the `additional_host_names` channel option is used to specify the names of additional hosts, or additional destination domain names, with which the channel communicates.

In legacy configuration, each such additional host name is specified as the name on the third or subsequent lines of a channel definition:

```
tcp_local ...keywords...
tcp-daemon
```

```
special1.domain.com  
special2.domain.com  
...etc...
```

33.3.15 Incoming channel match and switch channel options

Many MTA configuration choices look at the incoming channel (source channel) for a message. A number of channel options affect such initial channel "matching" and channel "switching".

33.3.15.1 Selection of alternate source channels (`switchchannel`, `allowswitchchannel`, `noswitchchannel`, `userswitchchannel`)

When an MTA server process (*e.g.*, an SMTP server process) accepts an incoming connection from a remote system it must choose a so-called source channel with which to associate the connection. Normally this decision is based on the transport used; for example, an incoming TCP/IP connection to an SMTP server (a Dispatcher service running the SMTP server image) is automatically associated with the `tcp_local` channel, while an incoming TCP/IP connection to an [LMTP server](#) (a Dispatcher service running the LMTP server that delivers to the Message Store) is automatically associated with the `tcp_lmtpss` channel - unless such automatic, default associations are overridden via the Dispatcher's [parameter option](#) being set to a `CHANNEL=channel-name` value, *e.g.*:

```
msconfig> ! Do not need to set the dispatcher.service:SMTP_SPECIAL.image  
msconfig> ! since the service name begins with "SMTP"  
msconfig> set dispatcher.service:SMTP_SPECIAL.logfilename IMTA_LOG:tcp_special_server.log  
msconfig# set dispatcher.service:SMTP_SPECIAL.tcp_ports special-port  
msconfig# set dispatcher.service:SMTP_SPECIAL.parameter "CHANNEL=tcp_special"
```

(Note the quoting of the value of the `parameter` option, required due to the presence of the equal sign character, =.)

In legacy configuration, this corresponds to a definition in the `dispatcher.cnf` file along the lines of:

```
[SERVICE=SMTP_SPECIAL]  
IMAGE=IMTA_BIN:tcp_smtp_server  
LOGFILE=IMTA_LOG:tcp_special_server.log  
PORT=special-port  
PARAMETER=CHANNEL=tcp_special
```

In the case of SMTP, however, the convention of a single incoming channel breaks down when it is desired to handle connections from different sources differently. That is, in the SMTP case, it can be useful to have the association of source channel with connections be controlled not only at the Dispatcher level, but for further distinctions (so-called source channel "switching") to be performed based on additional criteria, within the SMTP server process itself.

The `switchchannel` channel option provides a way associate different incoming channels with different source IP addresses. If `switchchannel` is specified on the initial channel the SMTP server uses, the IP address of the connecting (originating) host will be rewritten (using envelope From style rewriting) and matched against the channel table and if it matches then

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). `noswitchchannel` 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 will have no effect. At present `switchchannel` only affects SMTP channels, but there are actually no other channels where `switchchannel` would be reasonable. In particular, internal channels like `conversion` or `process` never need to switch. Also, LMTP servers do not support `switchchannel`.

Note that use of the (not-recommended) [CHECK_SOURCE TCP/IP-channel-specific option](#) setting with a value of 0 effectively disables `switchchannel` from taking effect.

The (new in JES MS 6.3-0.15) `userswitchchannel` channel option, if enabled on the current SMTP source channel, allows channel switching based on the envelope From address. If the envelope From address after rewriting is that of a user in the LDAP directory, then a per-user LDAP attribute (the attribute named by the [ldap_source_channel MTA option](#)), or if the user has no such attribute a per-domain LDAP attribute (the attribute named by the [ldap_domain_attr_source_channel MTA option](#)) will be consulted to find the name of a channel to which to switch as the new source channel. The source channel may change to any channel marked `switchchannel` or `allowswitchchannel` (the default).

Note that since the `userswitchchannel` plus user-or-domain LDAP attribute based switching is being done based on the envelope From (MAIL FROM) address and since such addresses are easily forged, this functionality should be used with great care. It is provided as a convenience for achieving esthetic and convenience features in the handling of messages purportedly from particular users or domains; but `userswitchchannel` does not provide the security of switching based on source IP address (`switchchannel`) or on authenticated sender information ([saslswitchchannel](#) and even more so the `mailSMTPSubmitChannel` LDAP attribute). When truly secure user-based channel switching is desired, instead the use of SMTP AUTH should be enforced and `mailSMTPSubmitChannel` used. (As of the 8.0 release, the LDAP attribute of relevance may be named something other than `mailSMTPSubmitChannel`, as specified via the [ldap_auth_attr_submit_channel MTA option](#).)

See also the [FROM_ACCESS mapping table's](#) `$~` flag which provides another way to do source channel "switching" (particularly well suited for "switching" the source channel for incoming notification messages).

33.3.15.2 Channel switching based on SMTP authentication (`saslswitchchannel`, `nosaslswitchchannel`)

The `saslswitchchannel` channel option is used to cause incoming connections to be switched to a specified channel upon a client's successful SASL use. (See the [maysasl*](#) and [mustsasl*](#) channel options for configuration of permitting/requiring SMTP AUTH and SASL use.) `saslswitchchannel` takes a required value, specifying the channel to which to switch. `nosaslswitchchannel` is the default, and means that channel switching is not performed upon a client's successful SASL use.

See also the `mailSMTPSubmitChannel` user LDAP attribute, (or as of the 8.0 release, whatever LDAP attribute is named by the [ldap_auth_attr_submit_channel MTA](#)

option) which when set on a user entry will cause channel "switching" to the specified channel; it thus permits "finer-grained" channel switching than `saslswitchchannel` which merely switches all authenticated submissions to a particular named channel.

See also the (new in JES MS 6.3) `userswitchchannel` channel option which, in conjunction with site-selected user or domain LDAP attributes, also allows "fine-grained" channel switching, in this case based merely on the *purported* From: address.

The `saslswitchchannel` channel option is typically used when it is desired to distinguish between authenticated *vs.* unauthenticated submissions as a class; the `mailSMTPSubmitChannel` user LDAP attribute (or as of the 8.0 release, whatever LDAP attribute is named by the `ldap_auth_attr_submit_channel` MTA option) is typically used when it is desired to securely distinguish submissions from particular users (say to allow "special privileges" to particular users); the (new in JES MS 63) `userswitchchannel` channel option and associated LDAP attribute(s) are typically used when it is desired to make esthetic distinctions (rather than more critical "secure" distinctions) on users' submissions without requiring authenticated verification of the sender address.

See also [Blocking SMTP relaying](#) for an example of typical use of `saslswitchchannel`.

Note that any channel switching done by `saslswitchchannel` will be undone if/when a client issues a (nonstandard, new in 8.0) XUNAUTHENTICATE command. (SMTP server support for the nonstandard XUNAUTHENTICATE extension and associated XUNAUTHENTICATE command is new in 8.0; note that XUNAUTHENTICATE is not supported for the LMTP server. XUNAUTHENTICATE is only valid after successful authentication has been performed, and the capability only shows up in the EHLO response at this point at well. Successful execution of the XUNAUTHENTICATE command will return the SMTP session to an unauthenticated state.)

33.3.15.3 Transport Layer Security (`maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, `tlsswitchchannel`)

The `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `notls`, `notlsclient`, `notlsserver`, and `tlsswitchchannel` channel options are used to configure STARTTLS use during the SMTP protocol by SMTP based channels such as [TCP/IP channels](#).

Note that prior to 7.0.5, the [LMTP server](#) did not support TLS use; as of 7.0.5, the LMTP server does support TLS, configured via the same `maytls`, `maytlsserver`, `musttls`, `mustlsserver`, channel options used to configure SMTP server TLS support.

`notls` is the default, and means that STARTTLS will not be permitted or attempted. It subsumes the `notlsclient` channel option, which means that TLS use will not be attempted by the SMTP/LMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections) and the `notlsserver` channel option, which means that TLS use will not be permitted by the SMTP/LMTP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP/LMTP 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 SMTP/LMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP/LMTP server that supports TLS, and `maytlsserver`, which means that the SMTP/LMTP server

will advertise support for the STARTTLS extension and will allow TLS use when receiving messages. Note that `maytls*` settings mean that the MTA *will* want to use TLS with remote sides that support STARTTLS, while allowing remote sides that do not have STARTTLS support to communicate without TLS; but `maytls*` settings do *not* inherently mean that the MTA will "fall back" to non-TLS use when TLS negotiation is attempted but fails: failure of TLS negotiation will result in that connection being closed as a failed connection (recorded with an "X" record). As of 8.0, with `maytlsclient` set, the MTA's client will attempt a new connection to attempt sending without TLS in cases where the remote SMTP server advertised TLS support but where the actual TLS negotiation failed; prior to 8.0, a failure in the TLS negotiation would immediately abort the delivery attempt for the message.

Specifying `musttls` will cause the MTA to insist upon TLS in both outgoing and incoming connections; e-mail will not be exchanged with remote systems that fail to successfully negotiate TLS use. It subsumes `musttlsclient`, which means that the SMTP/LMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP/LMTP servers that do not successfully negotiate TLS use (the MTA will issue the STARTTLS command and that command must succeed), and `musttlserver`, which means that the SMTP/LMTP 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. When `musttls` or `musttlserver` is on a channel, then unless TLS has been successfully negotiated all MAIL FROM: attempts will be rejected with the error:

```
530 5.7.0 No STARTTLS command has been given.
```

The `tlsswitchchannel` channel option is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. (This includes either successful STARTTLS use on a "regular" port, or use of the deprecated approach of negotiating upon connection to a "dedicated to TLS" port, usually port 465, configured via the Dispatcher's `ssl_ports` option in Unified Configuration, or its `TLS_PORT` option in legacy configuration.) `tlsswitchchannel` takes a required value, specifying the channel to which to switch.

Note that TLS library initialization is performed for any SMTP/LMTP channel which has any TLS usage permitted (or required). In particular, TLS library initialization will be performed by the TCP client for a channel marked merely `maytlserver`. (This overhead is normally fairly negligible.)

Note that these options affect only TLS use negotiated at the SMTP protocol level via STARTTLS; they do not affect potential TLS use triggered by connection to a port dedicated to TLS use such as with the `ssl_ports` Dispatcher service option.

33.3.16 Logging and debugging channel options

Basic channel logging of message transactions, and basic channel debugging, are controlled by several channel options.

For greater fine-tuning of exactly what is logged in transaction log entries, see [Transaction logging MTA options](#). For greater detail in debug log files, see [Debug MTA options](#).

33.3.16.1 Message transaction logging (`logging`, `nologging`, `logheader`)

The MTA provides facilities for logging each message as it is enqueued and dequeued. All log entries are made to the [MTA message transaction log file](#) `mail.log_current` in the MTA log directory, (*i.e.*, `msg-install-path/data/log/mail.log_current`). Message transaction logging is controlled on a per-channel basis. The `logging` channel option activates message transaction logging for a particular channel while the `nologging` channel option disables it. Logging is disabled on all channels by default although most default configurations enable logging on all channels.

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. Note that the MTA itself never does anything to the cumulative `mail.log` file and it is up to each site to manage (*e.g.*, delete, truncate, backup, *etc.*) that log file however they choose.

The [MTA message transaction log file](#) is written as a normal ASCII text file, whose exact format is configurable via the `log_format` MTA option.

If you wish to have all of your channels log message activity to the logging file, then simply add the `logging` channel option to a defaults channel.

When message transaction logging is enabled, the (new in JES MS 6.0) `logheader` option may be used on channels to additionally enable logging of message headers. It takes an encoded integer argument, where bit 0 (value 1) causes logging during both enqueue and dequeue operations, while a value of 2 causes logging during enqueue operations without enabling logging for message dequeues. The specific header lines to be logged are controlled by the `log_headers.opt` file, discussed along with the MTA option [log_header](#).

33.3.16.2 Debugging channel master and slave programs (`master_debug`, `nomaster_debug`, `slave_debug`, `noslave_debug`)

Some channel programs include optional code to assist in debugging by producing additional diagnostic output. Two channel options are provided to enable generation of this debugging output on a per-channel basis. The options 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 MTA 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`.

For the UNIX L and native channels, the "normal" debugging when the L channel is delivering is caused by `slave_debug`.

For the [reprocess channel](#), since it performs some of its functions/checks as if it "were" the prior channel (the channel that enqueued to the `reprocess` channel), to get debugging of the `reprocess` channel's enqueueing operation, you will need to enable `master_debug` on that prior channel, rather than on the `reprocess` channel itself.

Note that the [TCP/IP channel](#) master program will produce multiple `tcp_y_master.log` files per master channel program execution when `master_debug` is enabled. The first such

file produced shows the channel's determination of how many outgoing threads to start up; an additional log file will be created for each individual outgoing thread.

The Message Store delivery channels, `ims-ms` and `tcp_lmtpss*` ([LMTP back end TCP/IP channel](#), that is, the LMTP server), have two (or three) types of debug output: the MTA type of debug output (enabled as normal for MTA channels via the MTA channel options discussed here, and going to MTA channel debug log files as normal), plus some Message Store injection debugging (which is enabled by the `loglevel` option for the MTA, and which goes to the log file named, literally, `imta`, plus if message tracing is enabled (the `activate` `message` option is enabled) more detailed Message Store message tracing (which goes to the log file named, literally, `msgtrace`).

Not all MTA channel programs have debugging support code, and the amount of debugging output available also varies among those channel programs that include debug support.

33.3.17 Long address lists or headers channel options

Sites may desire to configure special handling of messages with many recipient addresses, or excessively long header lines. There are a number of channel options relating to such special handling.

33.3.17.1 Triggering alternate channel processing

**(`alternatechannel`, `alternateblocklimit`,
`alternatelinelimit`, `alternaterecipientlimit`)**

It is sometimes useful to force processing of messages meeting certain criteria to occur on a channel distinct from the one chosen by the MTA's alias expansion and rewriting process. The `alternatechannel` channel option provides a means to specify such a channel while the `alternateblocklimit`, `alternatelinelimit`, and `alternaterecipientlimit` channel options specify the criteria for when the alternate channel should be used.

`alternatechannel` takes the name of the alternate channel to use as an argument. `alternateblocklimit` takes an unsigned integer as an argument; the alternate channel will be used if the computed block size of the message exceeds this value. `alternaterecipientlimit` also takes an unsigned integer argument; the message will be queued to the alternate channel if the number of recipients queued to the current channel exceeds this value. Finally, the `alternatelinelimit` channel option takes an unsigned integer argument; the alternate channel will be used if the computed number of lines in the message exceeds this value.

Note that `alternaterecipientlimit` is a limit on envelope recipients for this message copy, on this channel; it has nothing to do with how many addresses may or may not be in the header; and envelope recipients on other channels are also irrelevant. However, the `alternaterecipientlimit` check does get performed before any message copy split-up due to storage of recipients per file controls such as [`addrisperfile`, `single`, or `single_sys`](#) channel option application.

Note that any [*SEND_ACCESS](#) or [*MAIL_ACCESS](#) mapping table probes will use the "original" destination channel name, not the alternate destination channel name, but a [CONVERSIONS](#) mapping table probe will use the alternate destination channel name.

Note that the alternate channel selection process is neither iterative nor recursive: Once an alternate channel has been selected it will be used regardless of what the various alternate channel options on that channel say to do.

33.3.17.2 Maximum allowed recipients or bad commands (`recipientlimit`, `recipientcutoff`, `deferralrejectlimit`, `disconnectrecipientlimit`, `disconnectrejectlimit`, `disconnectbadcommandlimit`, `disconnectbadburllimit`, `disconnectcommandlimit`)

The `recipientlimit`, `recipientcutoff` and (new in JES MS 6.2) `deferralrejectlimit` channel options, when placed on a source channel, impose per-channel limits on the number of recipients for a submitted message. Each of these options accepts a single integer argument; they default to no limit. (Note that setting `recipientlimit` or `recipientcutoff` to 0 has no effect; only positive limit values will be enforced.) These options are all per-channel (as opposed to per-SMTP-server) analogues of the [ALLOW_RECIPIENTS_PER_TRANSACTION](#), [REJECT_RECIPIENTS_PER_TRANSACTION](#), and [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) SMTP channel settings.

`recipientlimit` limits the total number of recipient addresses that will be accepted for the message; additional recipients will be rejected. The text in the rejection is configurable via the [error_text_recipient_over](#) MTA option, which by default is "too many recipients specified". The error is a temporary rejection by default, but if bit 4 (value 16) of the [use_permanent_error](#) MTA option is set, then the rejection is permanent. So in the case of attempted SMTP message submissions, the default temporary error, with the default error text, would appear as:

```
451 4.5.3 too many recipients specified
```

whereas with the default `error_text_recipient_over` text but with bit 4 (value 16) of the `use_permanent_error` MTA option set, then a permanent error would appear as:

```
550 4.5.3 too many recipients specified
```

`recipientcutoff` compares the total number of recipients that were presented to the MTA to the specified value and a message will not be accepted for delivery to *any* recipients if the value is exceeded. In the case of attempted SMTP submissions, the message will be rejected at the DATA command with the SMTP rejection:

```
451 4.4.5 Error ending envelope - Too many recipients specified for this message
```

New in JES MS 6.2, and supported only for SMTP (not for LMTP), `deferralrejectlimit` limits the number of bad (failing) addresses that will be allowed during a single session; after this number, all subsequent recipient addresses, good or bad, will be rejected with a temporary error. In the case of attempted SMTP submissions, additional RCPT TO: commands will be rejected with the error:

```
451 4.5.3 Too many rejections; try again later
```

while attempted VRFY: commands will be rejected with the error:

451 4.5.3 Verification blocked; too many rejections

Similar limits controlled on a per-user and per-domain basis can be configured via LDAP attributes; see the MTA options [ldap_recipientlimit](#), [ldap_recipientcutoff](#), [ldap_domain_attr_recipientlimit](#) and [ldap_domain_attr_recipientcutoff](#).

The `disconnect*` channel options are new in Messaging Server 6.2 (except: `disconnectcommandlimit` new in Messaging Server 7.1, `disconnectbadburllimit` new in Messaging Server 7.4-18.01). They are supported only for SMTP, not for LMTP. Each takes an integer argument specifying the maximum number of (recipients, rejections, bad commands, or commands, as applicable) which will be accepted for messages submitted via that source channel; any more will result in the MTA forcing a disconnect of the SMTP session, after issuing an error response to the client consisting of (for `disconnectrecipientlimit`):

421 4.7.0 Session recipient limit reached; disconnecting

for `disconnectrejectlimit` in JES MS 6.2:

450 4.7.0 Session bad recipient limit reached; disconnecting

or in JES MS 6.3 and later (JES MS 6.3 also being when behavior was enhanced so that rejected MAIL FROM's count against the `disconnectrejectlimit`, whereas in JES MS 6.2 only rejections at the RCPT TO or VRFY stages counted; JES MS 6.3 is also when behavior was enhanced so that `disconnectrejectlimit` is checked at the VRFY stage and with a negative value applied at the VRFY stage, whereas previously such VRFY rejections were counted but the disconnect would not be triggered until a subsequent RCPT TO attempt "noticed" that the threshold was exceeded):

421 4.7.0 Session rejection limit reached; disconnecting

or (for `disconnectcommandlimit`):

450 4.7.0 Maximum number of commands exceeded

In the case of the `disconnectrecipientlimit` or `disconnectrejectlimit` channel options, once the limit is exceeded, the error-response-and-disconnect normally will occur after the next MAIL FROM or RSET command (or in the case of `disconnectrejectlimit` in JES MS 6.3 and later, potentially after a failed VRFY attempt). (Note that because the disconnect usually does not happen until after a subsequent MAIL FROM or RSET, these `disconnect*` channel options would most often be used in conjunction with other channel keywords or [TCP/IP-channel-specific option](#) settings: perhaps `recipientlimit` or `recipientcutoff` to limit the number of recipient addresses accepted, or `deferralrejectlimit` or the [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) TCP/IP-channel-specific option setting.) In the case of `disconnectbadcommandlimit`, `disconnectbadburllimit`, and `disconnectcommandlimit`, once the limit is exceeded the error response is issued and the MTA forces the disconnect.

The `disconnectbadburllimit` channel option is new in Messaging Server 7.4-18.01. A single integer parameter is accepted specifying the number of invalid [BURL commands](#) that will be allowed before disconnecting. The default is 3.

Note that VRFY attempts are counted separately from RCPT TO attempts against the recipient count; that is, one may have up to the recipient limit of VRFYs *and* up to the recipient limit of RCPT TOs. The VRFYs are counted, but counted separately from RCPT TOs. In contrast, failed VRFY attempts are added to the same rejection counter used for counting failed RCPT TO attempts and MAIL FROM attempts for purposes of comparison against the rejection limit; that is, one may have up to the rejection limit of any combination of failed VRFYs, MAIL FROMs, or RCPT TOs.

When the `deferralrejectlimit` has been reached (or a TCP/IP-channel-specific option setting of `ALLOW_REJECTIONS_BEFORE_DEFERRAL` has been reached), a client VRFY attempt will receive from the MTA an error response:

```
451 4.5.2 Verification blocked; too many rejections
```

Note that prior to JES MS 6.3, the error was instead:

```
452 4.5.2 Verification blocked; too many bad addresses.
```

Note that the `FROM_ACCESS` mapping table's `$$S` flag may also be used to set limits such as `recipientlimit` or `recipientcutoff`.

For forcing IMAP or POP disconnection after a specified number of protocol errors -- similar to the `disconnectbadcommandlimit` effect for SMTP -- see the `maxprotocolerrors` IMAP or POP option.

33.3.17.3 Expansion of multiple addresses (`expandlimit`, `expandchannel`, `holdlimit`)

Most MTA 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 ("on-line" delays). In particular, multi-recipient messages that require a great deal of processing of the message body can be affected by processing delays, or that require creation of many different file copies on disk in the MTA queue area can be affected by slow disk performance. If the resulting delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred ("off-line") processing of additional recipient message copies once a given number of addresses are specified for a single message. The deferral happens after processing of the "initial" recipients (those before the `expandlimit` value was reached), and after address processing for the additional recipients too, (for instance, after `*_ACCESS mapping table` checks and after alias processing), but before message processing. In particular, such deferral means that for the "additional" recipients, only *one* message file (storing all of the "additional" recipients), is written to the queue area (to a `reprocess*` or `process*` channel queue area, depending upon use of the `expandchannel` channel option). 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` channel option. The `expandlimit` option takes an integer argument that specifies at what number of recipients to begin deferring processing

of the message copy (or copies) to that and additional recipient addresses. The default value is effectively infinite if the `expandlimit` channel option is not specified. A value of 1 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` channel option 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` channel option; 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 some purposes. In particular, for Sun ONE Messaging Server MTA versions 5.2 and earlier, typical configuration usage required that a processing channel, rather than a reprocessing channel, be used. 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` option to have any effect. If your configuration was built by the initial configuration utility, then you should already have a reprocessing channel.

(Note that typical Messaging Server sites running version 5.2 or earlier could not use the `expandlimit` option unless they also marked the affected channel `expandchannel process` (or `expandchannel process_somethingorother` redirecting the expansion to an alternate `process_*` sort of channel), as enqueues to a channel marked `viaaliasrequired` would not succeed if deferred to a `reprocess*` channel.)

Extraordinarily large lists of recipient addresses are often a characteristic of so-called SPAM--junk e-mail. The `holdlimit` channel option 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` channel option). As `.HELD` messages, the files will sit unprocessed in that MTA queue area awaiting manual intervention by the MTA postmaster.

33.3.17.4 Specify maximum length header line that the MTA will rewrite (`maxprocchars`)

Processing of long header lines containing lots of addresses can consume significant system resources. (Note, however, that resource consumption is much reduced in PMDF V5.0 as compared with previous versions of PMDF.) The `maxprocchars` source channel option is used to specify the maximum length header line that the MTA will process and rewrite. Messages with header lines 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 to process header lines of any length.

33.3.18 Message hash channel options

Message hash channel options are typically used in conjunction with a message archiving configuration, and hence with the [Message archival and hashing MTA options](#).

33.3.18.1 Message hashes (`deletemessagehash`, `generatemessagehash`, `keepmessagehash`)

The `generatemessagehash`, `keepmessagehash`, and `deletemessagehash` destination channel options control whether the MTA generates, retains, or deletes a message hash, respectively. All of these channel options first appeared in the 6.3 release. A message hash, if present, is stored in a `Message-hash: header` line. `deletemessagehash` is the default.

These keywords are intended to be used in conjunction with message archiving, along with [message archiving MTA options](#), and [Message Store archive options](#).

Note that AXS:One archiving will generate its own message hash on the messages it receives, if the MTA has not already generated and inserted such a message hash. However, in order to correlate messages in the Message Store with the messages archived by AXS:One, it is necessary that the MTA generates the hash and that that hash is retained in the messages delivered to the Message Store. Therefore, channels that deliver to the Message Store (`ims-ms` or `tcp_lmtpcs*` sorts of channels) should be marked either `generatemessagehash` or (if the message hashes are being initially generated at an earlier channel stage) `keepmessagehash`. In contrast, a channel delivering to remote Internet hosts, such as typically the `tcp_local` channel, would typically be marked `deletemessagehash` even if messages going out `tcp_local` are being archived, since there would in general be no expectation that the remote Internet sites would do anything (get any benefit) from the message hashes.

The choice of which channel(s) to mark with `generatemessagehash` will depend upon how "soon" *vs.* "late" one wishes to choose to consider a message's essential characteristics to be established. Generating a hash "early" (as soon as a message first comes into the MTA) may be before various changes (*e.g.*, changes performed by the conversion channel) occur, or before "split up" of a message into separate copies for different classes of recipients. More "different" (in one way or another) "copies" of a message may be considered "the same" for the archival purposes when hash generation is performed "early". Whereas when hashing is performed "late" (at the final delivery channel stage), then additional sorts of differences in message "copies" can be reflected in the message hash value (more distinctions between "copies" occur). But this may be either a plus or a minus depending upon site goals.

33.3.19 Message tracking channel options

New in the 8.0 release, a general [message tracking and recall facility](#) has been implemented. There are a number of [channel options](#) relating to message tracking.

33.3.19.1 Message Tracking and Recall Channel Options (`nottrackingclient`, `nottrackingserver`, `trackingclient`, `trackingdelivered`, `trackingfirst`, `trackinginternal`, `trackingmultiple`, `trackingrelayed`, `trackingserver`, `trackingsingle`, `trackingtimeoutdefault`, `trackingtimeoutmax`, `trackingtimeoutmin`)

The message tracking and recall facility consists of an SMTP service extension ([RFC 3885](#)) as well as a separate MTQP server ([RFC 3887](#)). (Note that these standards only specify message tracking; message recall is an Oracle extension.) Mail clients can use this facility to track and possibly recall messages they have sent. This set of keywords controls the availability and handling of the Message Tracking SMTP extension.

The extension is enabled with the `trackingserver` source channel option. The `nottrackingserver` channel option disables the availability of the extension, and is the

default. The `trackingtimeoutdefault` source channel option specifies the default timeout in seconds if no timeout value is specified in the MTRK command, as allowed by the protocol. The default is 3 days.

The `trackingtimeoutmin` and `trackingtimeoutmax` source channel options specify the minimum and maximum allowed timeout value in seconds; any value greater than the maximum or less than the minimum is silently lowered or raised to the corresponding limit. The defaults are 1 and 14 days, respectively.

The SMTP client's use of the extension is controlled by the `trackingclient` and `nottrackingclient` channel options. The former enables use of the extension; the latter disables it and is the default. Note that the extension must be enabled on clients and servers throughout a deployment in order for tracking and recall to work across that deployment.

The handling of messages relayed internal to a deployment, including internal channel hops, needs to be distinguished from the case where messages leave the administrative domain for tracking and recall to work properly. However, the SMTP protocol is commonly used in both cases. Additionally, the case where a successful channel dequeue results in message delivery also needs to be distinguished from dequeues where this does not occur.

Three channel options are provided to specify these semantics: `trackinginternal`, `trackingrelay`, and `trackingdelivered`. `trackinginternal`, the default, specifies that the message is being transferred internally. `trackingrelayed` specifies that the channel transfers messages to some external system. Finally, `trackingdelivered` specifies that the channel performs final delivery of the message.

[RFC 3885](#) specifies how the Message Tracking SMTP Extension interacts with aliases and mailing lists. In particular, it says, "MTAs MUST NOT copy MTRK certifiers when a recipient is aliased, forwarded, or otherwise redirected and the redirection results in more than one recipient. However, an MTA MAY designate one of the multiple recipients as the "primary" recipient to which tracking requests shall be forwarded; other addresses MUST NOT receive tracking certifiers. MTAs MUST NOT forward MTRK certifiers when doing mailing list expansion."

This arguably makes sense for tracking-only applications where presenting the results of a complex alias expansion process to the end user may be confusing; however, the situation with message recall is different. Users expect recall to work when feasible, including when alias expansion is involved. (Mailing lists are different; a mailing list effectively "owns" its messages once it expands, so recall past a mailing list expansion is inappropriate.)

Accordingly, three source channel options are provided to control the MTA's behavior in this regard. `trackingmultiple`, the default, tells the MTA to pass tracking id/timeout information to all recipients of an alias expansion. `trackingsingle` causes tracking id/time information to pass through only when there is a single recipient. Finally, `trackingfirst` causes tracking information to pass through to the first alias expansion recipient. (Note that in the case of aliases stored in LDAP, the first recipient is unpredictable.)

33.3.19.2 Automatic Tracking ID Generation (`trackinggenerate`)

Message tracking and recall depends on the generation, attachment, and transfer of tracking identifiers. Such identifiers are normally generated and attached to messages by the submitting client. However, essentially no clients currently support the generation of such identifiers, making it impossible to write a separate tracking/recall client to deal with messages submitted by a non-tracking-enabled client. Additionally, a user who elects to use multiple

clients, some tracking-enabled and some not, will end up with only a subset of their messages able to be tracked and recalled.

Tracking identifiers can also provide, independent of their use for user tracking and recall, a stable identifier that ties MTA log entries across multiple systems together in ways that envelope ids and `message-id` header fields do not and cannot. As such, automatic assignment of tracking identifiers to message on ingress as well as submission has real utility independent of user tracking and recall functions.

The `trackinggenerate` source channel addresses these needs. A single required integer parameter specifies the default tracking timeout. If set, a tracking identifier for the message is generated in one of two ways:

- If authentication has been used and the user has a general recall secret associated with their LDAP entry (see the `ldap_auth_attr_recall_secret` MTA option), then a per-message recall secret is generated by computing a SHA-1 hash of the concatenation of the content of `Message-id:` header field, the `Date:` header field (if present), and the user's general recall secret. The per-message recall is then hashed twice with SHA-1 to create the tracking identifier. The tracking timeout is controlled by the `trackinggenerate` value.
- If authentication wasn't used or no general recall secret is associated with the account, a tracking identifier is created by hashing a unique identifier with an MD4 hash. Note that the security of this process is controlled not by the randomness of the unique identifier or the use of MD4, but rather by the infeasibility of computing X given T , where $\text{SHA1}(X) = \text{MD4}(T)$.

33.3.20 MLS channel options

RESTRICTED: Not yet fully implemented. There are a few channel options relating to MLS (Multi Layer Security).

33.3.20.1 MLS (Multi Layer Security) Channel options: `m1slabel` (string), `m1srange` (string)

RESTRICTED: Not yet fully implemented.

33.3.21 Notification messages and postmaster messages channel options

[Notification message](#) handling, especially notification message generation, is an important function of the MTA with therefore a number of related channel options. See also the [Notification message MTA options](#).

33.3.21.1 Postmaster address recognition (`aliaspostmaster`)

Specifying the `aliaspostmaster` option on a destination channel causes any messages addressed to the username "postmaster" (lowercase, uppercase, or mixed case) at the official channel name to be redirected to `postmaster@local-host`, where `local-host` is the official local host name (the `official_host_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 will receive mail. So this setting can be useful when a site wants to centralize postmaster responsibilities, rather than having [separate postmaster accounts for separate domains](#). That is, whereas the `returnaddress` channel option controls what return

postmaster address is used when a notification message is generated *from* the postmaster, `aliaspostmaster` affects what is done with messages addressed *to* the postmaster.

33.3.21.2 Returned messages (`sendpost`, `nosendpost`, `copysendpost`, `errsendpost`)

A channel program may be unable to deliver a message due to long-term service failures or invalid addresses. When this happens the MTA channel program returns the message to the sender with an accompanying explanation of why the message was not delivered. The MTA will also optionally send a copy of certain failed messages to the [local postmaster](#). This is useful for monitoring message failures, but it can result in lots of traffic for the postmaster to deal with.

The options `sendpost`, `copysendpost`, `errsendpost`, and `nosendpost` are used to control the sending of failed messages to the postmaster. `sendpost` tells the MTA to send a copy of all failed messages to the postmaster unconditionally. `copysendpost` instructs the MTA to send a copy of the failure notice to the postmaster unless the originator address on the failing message is blank; *i.e.*, the postmaster gets copies of all failed messages except those messages that are actually themselves reporting on bounces or other notifications. `errsendpost` instructs the MTA to only send a copy of the failure notice to the postmaster when the notice cannot be returned to the originator. No failed messages are ever sent to the postmaster if `nosendpost` is specified.

The default in releases prior to 7.3-11.01 was `copysendpost`. As of 7.3-11.01 the default has been changed to `nosendpost`.

33.3.21.3 Warning messages (`warnpost`, `nowarnpost`, `copywarnpost`, `errwarnpost`)

In addition to returning messages as undeliverable, the MTA sometimes sends warnings detailing messages that it has been unable to deliver for some period of time. This is generally due to timeouts based on the setting of the [notices](#) channel option, 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 will continue. In most cases they also contain the headers and possibly some additional content from the message in question.

The MTA will also optionally send a copy of certain warning messages to the local postmaster. This can be somewhat useful for monitoring the state of the various MTA queues, although it does result in lots of traffic for the postmaster to deal with.

The options `warnpost`, `copywarnpost`, `errwarnpost`, and `nowarnpost` are used to control the sending of warning messages to the postmaster. `warnpost` tells the MTA to send a copy of all warning messages to the postmaster unconditionally. `copywarnpost` instructs the MTA to send a copy of the warning to the postmaster unless the originator address on the as yet undelivered message is blank; *i.e.*, the postmaster gets copies of all warnings of undelivered messages except for those as yet undelivered messages that are actually themselves reports on bounces or notifications. `errwarnpost` instructs the MTA to only send a copy of the warning to the postmaster when the notice cannot be returned to the originator. No warning messages are ever sent to the postmaster if `nowarnpost` is specified.

If no `*warnpost` channel option is in effect the default is taken from any `*sendpost` channel option setting. If no such setting exists the default in releases prior to 7.3-11.01 was `copywarnpost`. As of 7.3-11.01 the default has been changed to `nowarnpost`.

33.3.21.4 Notification and disposition channels (`notificationchannel`, `dispositionchannel`)

New in 6.2: By default, the MTA always generates notification messages (Delivery Status Notifications) such as bounce messages, warnings of delayed delivery, *etc.*, and disposition messages (Message Disposition Notifications) such as "vacation" messages, through the [process channel](#). That is, the channel that needs to generate a DSN or MDN submits the DSN or MDN, respectively, to the process channel; and the process channel subsequently enqueues the DSN or MDN on to the appropriate outbound channel (back to the sender of the original message).

The `notificationchannel` channel option may be used on a channel to tell it to generate DSNs through the specified channel, rather than through the (default) process channel. It takes a required argument, which is the name of the channel to which to enqueue the newly generated DSNs; normally, this should be some `process_something` channel that a site defines for this purpose.

The `notificationchannel` channel option may be helpful when a channel is prone to generating an exceptionally large number of DSNs, or when it is desired to [handle DSNs generated by a particular channel in some special way](#), in combination with source specific rewrite rules.

The `dispositionchannel` channel option may be used on a channel to tell it to generate MDNs through the specified channel, rather than through the (default) process channel. It takes a required argument, which is the name of the channel to which to enqueue the newly generated MDNs; normally, this should be some `process_something` channel that a site defines for this purpose.

33.3.21.5 Including altered addresses in notification messages (`includefinal`, `suppressfinal`, `useintermediate`)

When the MTA generates a [notification message](#) (bounce message, delivery receipt message, *etc.*), there may be several forms of an address available to the MTA: the preserved "original" form of a recipient address (the ORCPT form--in principle, that form originally typed by the sending user, but in practice that true original form may not have been preserved and instead some "later", transformed version may be the earliest form preserved), the "recently" active form (referred to here as the "intermediate" form) corresponding to the form of the address prior to any most recent forwarding applied by the MTA, and an altered "final" form of that recipient address (as for instance a final form after forwarding is applied). The MTA always includes the original form (assuming it is present) in the notification message, since 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`, `suppressfinal`, and `useintermediate` channel options, as set on a channel generating a notification message, control whether the MTA also includes the intermediate or final form of the address. `includefinal` means to include the final form of the recipient address; `useintermediate` is the default, and means to include the intermediate address form rather than the final address form; `suppressfinal` causes the MTA to suppress the final and intermediate address forms, if an original address form is present, from notification messages.

See [Notification message format](#) for an example of a DSN to see where these options would affect notification format.

Including the intermediate form is normally useful, especially to the postmaster of the MTA system itself, as the original address form, while presumably recognizable to the original

sending user, may bear no obvious relationship to the form of address active by the time the MTA processed the message; in order to figure out what recipient on the MTA system was intended, something like the intermediate or even final address form may well be necessary. Suppressing the inclusion of any final or intermediate form of address entirely 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 notification messages.

Note that only some channels fully support the `useintermediate` channel option; for other channels (including all channels written using the API), the effect of `useintermediate` is merely to use the final address form, that is, it is effectively equivalent to `includefinal`.

See also the special use of a colon character, `:`, as the [leading character of an alias value](#) which provides an alias-specific, rather than channel-wide, version of the `useintermediate` effect.

33.3.21.6 Default language tag (language)

Certain MTA operations, especially those involving selection of textual content to send to users, may need to take language into account. For example, nondelivery notification text may be available in multiple languages and the [NOTIFICATION_LANGUAGE mapping](#) and [DISPOSITION_LANGUAGE mapping](#) can be used to select the appropriate language to use.

The MTA also supports use (and selection amongst) language-tagged values for various `mailAutoReply*` LDAP attributes used to construct vacation messages; see the discussion of the [ldap_autoreply_subject](#) and [ldap_autoreply_text*](#) MTA options in particular.

It is axiomatic that in order to make decisions based on language language tagging information must be available. Normally this information is derived from the message being processed, *e.g.*, from an `Accept-language:` header field, from a `Preferred-language:` header field, or even from the country code found in the `From:` address. However, not all messages contain such information.

The `language` channel option can be used to associate a default language with a particular source channel. A single string argument is required specifying a language tag. Messages originating from this channel which aren't tagged in any other way will be effectively tagged with this value. The default is not to assume any language tag on a per-channel basis.

33.3.21.7 SMTP DSN extension support (notary, refusenotary, nonotary)

The `notary` and (the `RESTRICTED`) `nonotary` channel options control whether client [TCP/IP channels](#) attempt to use the SMTP DSN extension (defined in [RFC 3461](#)). The `notary` channel option is the default on SMTP over TCP/IP channels.

The `nonotary` channel option, if set, disables the use of the SMTP NOTARY extension. Its use on SMTP client channels is `RESTRICTED`, and it is normally used only on LMTP client channels. Note that setting [lmtp](#) or an [lmtp_*](#) channel option on a channel implicitly sets `nonotary`.

New in 8.0.1 is the `refusenotary` channel option. This `RESTRICTED` option disables the DSN extension in the SMTP/LMTP client and additionally, the SMTP server. (The DSN extension is never offered by the LMTP server.)

33.3.21.8 Undeliverable message notification times (notices, nonurgentnotices, normalnotices, urgentnotices)

The `notices`, `nonurgentnotices`, `normalnotices`, and `urgentnotices` channel options control the amount of time an undeliverable message is silently retained in a given channel queue. The MTA is capable of returning a series of [warning messages](#) to the originator and, if the message remains undeliverable, the MTA will eventually return the entire message.

Different return handling for messages of [different priorities](#) may be explicitly set using the `nonurgentnotices`, `normalnotices`, or `urgentnotices` channel options. Setting values for the `notices` option is equivalent to setting those values for `nonurgentnotices`, `normalnotices`, and `urgentnotices`, so those values will be used for all messages. (In particular, if you wish to have, for example, an `urgentnotices` setting override a more general `notices` setting, then the `urgentnotices` option must appear *after* the `notices` option.)

This channel option's required argument is a list of up to five monotonically increasing positive integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the MTA option `return_units` is 0 or not specified in the MTA option file, or hours if the MTA option `return_units` is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (*i.e.*, bounced). When it attains any of the other ages, a warning notice is sent. (Note that when `return_units` is 0, so that the ages are interpreted as days, the ages to be exceeded are interpreted as full, twenty-four hour days; for instance, in order for a `notices` value of 1 to apply to a message, the message must have been tried already for a full twenty-four hours. For instance, if the return job, as is usual, is configured to run at thirty minutes after midnight each day, and if the initial `notices` value is 1, then a message originally submitted on the first of a month will not get a notification message generated until thirty minutes after midnight on the third of the month; consider that on the second of the month, the message had not yet been being tried for a *full* twenty-four hours.)

The default if no `notices` channel option is given is to use the `notices` setting for the local, 1, channel. If no setting has been made for the local channel, then the defaults 3, 6, 9, 12 are used meaning that warning messages are sent when the message attains the ages 3, 6, and 9 days (or hours) and the message is returned after remaining in the channel queue for more than 12 days (or hours).

If you wish to change the notification ages for all of your channels, then the simplest thing to do is to add a `defaults` channel block at the top of your configuration (assuming you don't already have one) and set the appropriate `*notices` options there.

33.3.21.9 Postmaster address (`returnaddress`, `noreturnaddress`, `returnpersonal`, `noreturnpersonal`)

By default, the Postmaster return address used when the MTA constructs [bounce or 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 "Internet Mail Delivery" for the Messaging Server and "PMDF e-Mail Interconnect" for PMDF. Care should be taken in the selection of the Postmaster address---an illegal selection may cause rapid message looping and pile-ups of huge numbers of spurious error messages.

The `return_address` and `return_personal` MTA options can be used to set the system default for the Postmaster address and personal name. Or if per channel controls are desired, the `returnaddress` and `returnpersonal` channel options may be used. And finally, a hosted-domain specific Postmaster address can be set using the LDAP attribute (normally `mailDomainReportAddress`) named by the `ldap_domain_attr_report_address` MTA option.

These channel options `returnaddress` and `returnpersonal` each take a required argument specifying the Postmaster address and Postmaster personal name, respectively. `noreturnaddress` and `noreturnpersonal` are the defaults and mean to use the default values, either defaults established via the `return_address` and `return_personal` MTA options, or the normal default values if such options are not set.

33.3.21.10 Postmaster returned message content (`postheadonly`, `postheadbody`)

When an MTA 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. Note, 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 system privileges if they so choose.

The channel options `postheadonly` and `postheadbody` are used to control what gets sent to the postmaster. `postheadbody` returns both the headers and the contents of the message. It is the default. `postheadonly` causes only the headers of the returned message to be sent to the postmaster.

33.3.21.11 Passing read receipt requests to the VMS MAIL mailbox (`OpenVMS`) (`readreceiptmail`)

Since the VMS MAIL mailbox does not support read receipts, by default PMDF "downgrades" read receipt requests into delivery receipt requests when delivering to the VMS MAIL mailbox. However, some clients---such as IMAP clients when the VMS MAIL mailbox is served out by an IMAP server---may have some read receipt support themselves, handled outside of the VMS MAIL message store itself; when a site has such clients in use it may be desired to "pass through" any read receipt requests rather than "downgrading" them to delivery receipt requests.

The `readreceiptmail` channel option, when placed on the `l` (lowercase "L") channel, causes PMDF on OpenVMS to pass through read receipt requests.

33.3.21.12 Delivery receipt request style (`reportboth`, `reportheader`, `reportnotary`, `reportsuppress`)

The `reportboth`, `reportheader`, `reportnotary`, and `reportsuppress` channel options control which sort, if any, of delivery receipt request the MTA constructs from "foreign" delivery receipt requests, such as for messages coming in to PMDF via PMDF-LAN, PMDF-MR, PMDF-X400, or PMDF-MB400 channels, or via the addressing channel. On OpenVMS, these keywords also control the interpretation of delivery receipt requests from VMS MAIL or PMDF MAIL via L or D channels. For PMDF-LAN, PMDF-MR, PMDF-X400, and PMDF-MB400 channels, these keywords also control which sort of delivery receipt request the MTA will convert into the respective "foreign" delivery receipt request. (In the case of PMDF-X400, note that the keyword on the `MIME_TO_X400` channel controls the behavior in both directions, to and from the X.400 world.) The current default is `reportheader` meaning to turn "foreign" delivery receipt requests into the old ad-hoc header style delivery receipt requests. `reportnotary` requests that only NOTARY⁸ style delivery receipt requests be generated; this may become the default in a future version. `reportboth` causes the MTA to generate both a header style and a NOTARY style delivery receipt request when seeing a "foreign" delivery receipt request; setting this may result in two delivery receipts from MTAs

that support both forms of delivery receipt request. `reportsuppress` causes the MTA to ignore (suppress) incoming "foreign" delivery receipt requests.

33.3.21.13 Blank envelope return addresses (`returnenvelope`)

The `returnenvelope` channel option takes a bitmask argument.

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 addresses. Note that the use of a blank address is mandated by [RFC 1123](#). However, some systems do not handle blank envelope from address properly 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. Again, this is used to accomodate incompliant systems that don't conform to [RFC 821](#), [RFC 822](#), or [RFC 1123](#).

Bit 2 (value = 4) controls whether or not the MTA checks that any (non-empty) envelope From address matches (rewrites to) an MTA channel.

Setting bit 3 (value = 8) is equivalent to setting the [mailfromdnsverify channel option](#): it controls whether or not the MTA checks that the domain in the envelope From address resolves in the DNS. That is, setting the bit causes the MTA to require that a DNS entry can be found corresponding to the domain in the envelope From address; but the type of DNS entry does not matter.

Setting bit 4 (value = 16) causes the MTA to enforce that if the envelope From address claims a local domain name, the envelope From address must correspond to a user address (user alias).

Bit 5 (value = 32) modifies the effect of bit 3 (value 8). When bits 3 and 5 are both set (value = 40), then a DNS query resulting in an authoritative `HOST_NOT_FOUND` response will be treated as a temporary error (a 450 error), rather than being rejected with a permanent rejection (a 550 error) as `mailfromdnsverify (returnenvelope 8)` would otherwise normally cause.

New in 8.0, bit 6 (value = 64) modifies the effect of setting bit 3 (value = 8) on domain validity checks. With both these bits set, if the domain in the MAIL FROM address corresponds to a null MX domain, that address will be rejected as invalid. That is, setting bit 6 causes the bit 3 domain check to also implement support for draft-delany-nullmx-01.txt.

Note that the [return_envelope MTA option](#) can be used to set an MTA system default for this sort of behavior.

33.3.22 Processing control and job submission channel options

There are a number of important channel options reflecting the fundamental nature of the channel, and interacting with the [Job Controller's](#) scheduling of messages.

33.3.22.1 Number of message files or addresses to handle per service job or file (`addrsperjob`, `filesperjob`)

The MTA normally creates one delivery service job per channel that needs service. This applies to both immediate service and periodic service jobs: when a message is initially sent

and immediate service is needed one job is created for each channel to which the message is queued, and when the MTA creates periodic jobs it normally creates one periodic job for each channel that needs service.

A single service job may not be sufficient to insure prompt delivery of all messages, however. In particular, PMDF-FAX messages may take a long time to deliver; if multiple FAX modems are available it is not sensible to use a single job and a single modem.

The `addrspersperjob` and `filesperjob` channel options can be used to cause the MTA to create additional service jobs. Each one of these options takes a single positive integer parameter which specifies how many addresses or queue entries (*i.e.*, 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 an option is equivalent to specifying a value of zero. The effect of these options is maximized; the larger number computed will be the number of service jobs that are actually created.

The `addrspersperjob` channel option computes the number of services jobs to start by dividing the total number of To: addressees in all entries by the given value. The `filesperjob` channel option 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` channel options and the specification of header-modifying actions in mailing lists. Also note that the `maxjobs` channel option places an upper bound on the total number of service jobs that can be created.

For example, if a message with 4 recipient addresses is queued to a channel marked `addrspersperjob 2` and `maxjobs 5` a total of 2 service jobs will be created. But if a message with 23 recipient addresses is queued to the same channel only 5 jobs will be created because of the `maxjobs` restriction.

Note that these channel options affect the creation of both periodic and immediate service jobs. In the case of periodic jobs the number of jobs created is calculated from the total number of messages in the channel queue. In the case of immediate service jobs the calculation is based only on the message being entered into the queue at the time.

Finally, note that the `addrspersperjob` option is generally only useful on channels that provide per-address service granularity. Currently this is limited to PMDF-FAX channels; there is no case where it is useful for the Messaging Server.

33.3.22.2 Service job execution deferral (`after`, `urgentafter`, `normalafter`, `nonurgentafter`, `secondclassafter`, `thirdclassafter`)

Service jobs are created to deliver messages; the creation of such jobs on an as-needed basis is managed by the MTA's [Job Controller](#). The creation of service jobs can be deferred using the `after` channel option, or deferred on a priority-sensitive basis using the other `*after` channel options. Note, however, that such deferral is seldom of interest given the modern Job Controller's internal, "smart" management of jobs.

Each `*after` option accepts an argument specifying either an absolute time at which to initiate a delivery job, or an amount of time to delay (a delta time). For the Messaging Server MTA, if the argument is an unsigned integer value, it is interpreted as an absolute time at which to initiate a delivery job, in GMT time zone; if the argument begins with the plus character, `+`, it is interpreted as the number of seconds by which to defer the execution of the

job - a delta time value. (Note that for PMDF, the argument syntax was different: all values were interpreted as delta time values.)

Historical note: For PMDF, deferred execution with a (typically small) delta time value was most often used to increase throughput (*e.g.*, as a result of cutting down on process creation overhead) for heavily used channels. By using `after` to introduce a slight latency in the creation of a service job, each such job had a window of time during which to "collect" all the messages sent to the channel in that time. Whereas otherwise a service job might handle only one (or at especially busy times perhaps two or three) messages, such use of `after` allowed a service job to handle larger numbers of messages. For channels with high connection or process activation overhead, this could result in higher overall throughput. But this is no longer the case for the Messaging Server MTA.

Separate deferral settings are allowed for messages with different effective priority values. The `urgentafter` channel option sets the delay for urgent messages, `normalafter` sets the delay for normal priority messages, and `nonurgentafter` sets the delay for non-urgent messages. `secondclassafter` and `thirdclassafter` set the delay for second-class and third-class (nonstandard priority levels below non-urgent) messages respectively. `after` sets the delay for all messages regardless of priority; setting it is equivalent to setting all of the other `*after` options to that same delay value.

33.3.22.3 Delivery retry intervals (`backoff`, `urgentbackoff`, `normalbackoff`, `nonurgentbackoff`)

Backoff options specify the frequency of message delivery retries when messages aren't successfully delivered the first time. These options all accept a series of intervals as arguments. The first interval specifies the time to wait before the first retry, the second specifies the time to wait for the second retry, and so on. The last value given specifies the time to wait for all subsequent retries. Up to eight intervals can be specified. Deliveries are attempted for a period of time specified by the `notices` channel option. Delivery will fail if successful delivery cannot be made within the time allowed by the last `notices` channel option setting.

Interval values use [ISO 8601 periodic time syntax](#):

```
P[yearsY][monthsM][weeksW][daysD][T[hourSH][minutesM][secondsS]]
```

years, months, weeks, days, hours, minutes and *seconds* are all integer values.

Separate interval settings are allowed for messages with [different priority](#) settings. The `urgentbackoff` channel option sets the retry intervals for urgent messages, `normalbackoff` sets the retry intervals for normal priority messages, and `nonurgentbackoff` sets the interval for non urgent messages. `backoff` sets the retry intervals for all messages regardless of priority.

The default intervals between delivery retries attempts in minutes are:

```
urgent: 30, 60, 60, 120, 120, 120, 240
normal: 60, 120, 120, 240, 240, 240, 480
nonurgent: 120, 240, 240, 480, 480, 480, 960
```

Note that [ims-ms channels](#) and [LMTP client TCP/IP channels](#) have special case handling that overrides normal `backoff` for the specific error condition of encountering `IMAP_MAILBOX_LOCKED` when attempting delivery to a Message Store user.

33.3.22.4 Initiating delivery processing (bidirectional, master, slave)

Three options 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 options is specified, is `bidirectional`. These options determine whether the MTA bothers to initiate delivery activity when a message is queued to the channel - there is no point in doing this on a slave-only channel.

The use of these options 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 options should be used.

Note that the [Job Controller configuration](#) should, normally, include definitions for what actual image(s) a channel should execute for `master` or `slave` mode operations, as appropriate for that channel. For instance, a channel which is `bidirectional` (and where both directions are truly used) should normally have both `master_command` and `slave_command` Job Controller options defined, whereas a channel which operates solely in `master` mode needs only a `master_command` option, not a `slave_command`.

33.3.22.5 Deferred delivery dates (deferredsource, nodeferredsource, deferreddestination, nodeferreddestination)

The `deferredsource` and `deferreddestination` channel options implement recognition and honoring of the Deferred-delivery: header. These options are newly implemented in the 7.0 release of Messaging Server. When set on a source or destination channel respectively, messages with a deferred delivery date in the future will be held in that channel queue until the deferred delivery date is reached. See [RFC 2156](#) for details on the format and operation of the Deferred-delivery: header line. Both channel options accept a single integer argument specifying the maximum number of seconds in the future a Deferred-delivery: date value can specify and still be honored. If both options apply to a transaction the lower of the two limits will apply. The integer arguments are optional in a traditional `imta.cnf` file - if the value is omitted a default of `60*60*24*7` (7 days) will be assumed.

Prior to the 7.0 release the `deferred` channel option provides similar functionality to `deferreddestination`. In 7.0 the MTA accepts `deferred` as a synonym for `deferreddestination` in the `imta.cnf` file; in Unified Configuration, `deferred` and `nodeferred` may no longer be used and the preferred `deferreddestination` and `nodeferreddestination` must be used instead. Note also that prior to 7.0 deferred delivery was not a reliable service. In particular, deferred handling information does not survive [Job Controller](#) restarts.

`nodeferredsource` and `nodeferreddestination` are the defaults. It is important to realize that while support for deferred message processing is mandated by [RFC 2156](#), actual implementation of it effectively lets people use the mail system as an extension of their disk quota.

See also the [\[DEFERRED\]](#) alias file named parameter and, in Unified Configuration, the `alias_deferred` alias option, which provide a per-alias (per-recipient) mechanism to add a Deferred: header line. For users defined in LDAP, the attribute `mgrpAddHeader`, or whatever attribute is named by the `ldap_add_header` MTA option, also provides a way to add such a header line.

See also the [futurerelease](#) channel option, which enables use of the SMTP FUTURERELEASE extension, offering superior functionality over Deferred-delivery: header line based message deferral.

33.3.22.6 Expansion of multiple addresses (`expandlimit`, `expandchannel`, `holdlimit`)

Most MTA 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 ("on-line" delays). In particular, multi-recipient messages that require a great deal of processing of the message body can be affected by processing delays, or that require creation of many different file copies on disk in the MTA queue area can be affected by slow disk performance. If the resulting delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred ("off-line") processing of additional recipient message copies once a given number of addresses are specified for a single message. The deferral happens after processing of the "initial" recipients (those before the `expandlimit` value was reached), and after address processing for the additional recipients too, (for instance, after [*_ACCESS mapping table](#) checks and after alias processing), but before message processing. In particular, such deferral means that for the "additional" recipients, only *one* message file (storing all of the "additional" recipients), is written to the queue area (to a [reprocess*](#) or [process*](#) channel queue area, depending upon use of the `expandchannel` channel option). 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` channel option. The `expandlimit` option takes an integer argument that specifies at what number of recipients to begin deferring processing of the message copy (or copies) to that and additional recipient addresses. The default value is effectively infinite if the `expandlimit` channel option is not specified. A value of 1 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` channel option 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` channel option; 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 some purposes. In particular, for Sun ONE Messaging Server MTA versions 5.2 and earlier, typical configuration usage required that a processing channel, rather than a reprocessing channel, be used. 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` option to have any effect. If your configuration was built by the initial configuration utility, then you should already have a reprocessing channel.

(Note that typical Messaging Server sites running version 5.2 or earlier could not use the `expandlimit` option unless they also marked the affected channel `expandchannel`.)

`process` (or `expandchannel process_somethingorother` redirecting the expansion to an alternate `process_*` sort of channel), as enqueues to a channel marked `viaaliasrequired` would not succeed if deferred to a `reprocess*` channel.)

Extraordinarily large lists of recipient addresses are often a characteristic of so-called SPAM--junk e-mail. The `holdlimit` channel option 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` channel option). As `.HELD` messages, the files will sit unprocessed in that MTA queue area awaiting manual intervention by the MTA postmaster.

33.3.22.7 SMTP Future Release Extension (`futurerelease`)

Release 7 of the Messaging Server MTA implements support for future release SMTP SUBMIT extension defined in [RFC 4865](#). This support is enabled by placing the `futurerelease` channel option on the source channel used for initial message submission. The option takes a single integer argument: The maximum number of seconds a message can be deferred.

Care should be used when enabling future release since it allows messages to be in effect stored in the MTA's queues. Future release should only be used for channels handling initial message submission and authentication should be required.

Note that similar functionality is available in earlier Messaging Server releases: Specification of a `Deferred-delivery:` header field in a submitted message coupled with use of the `deferred` channel option on the destination channel provided the ability to defer delivery of messages. However, future release provides superior functionality:

- The facility is controlled by a setting on the source channel, allowing it to be provided to a subset of the user population. Placing `deferred` on a destination channel opened the door to anyone submitting a message to that channel that would be deferred for some period of time.
- There's no way for a client which sets a `Deferred-delivery:` header field to know whether or not the header has actually caused the message to be deferred. The future delivery SMTP extension, on the other hand, lets the client know how long a message can be deferred and an error will be returned to the client if the message cannot be deferred for the time the client wants.
- There was no way to place a limit on the amount of time a message could be deferred. Instead what happened was that a message deferred longer than the channel's last `notices` value would simply be returned as undeliverable.
- `Deferred-delivery` settings on messages did not survive a Job Controller restart.

As part of the implementation work for future release the old `Deferred-delivery:` mechanism has been redesigned to address some (but not all) of these points. In particular, the `deferred` channel option has been replaced by two new channel keywords, `deferredsource` and `deferreddestination`. (The `deferred` option is now a synonym for `deferreddestination`.) Both of these options accept an integer argument (required in unified configurations, optional in `imta.cnf`) specifying in seconds the maximum amount of time in the future a `Deferred-delivery:` header can specify and still be honored. The default if no argument is specified is `60*60*24*7`, or 7 days. `deferredsource` enables `Deferred-delivery:` processing on the basis of the source channel while `deferreddestination` operates on destination channels. Finally, `Deferred-delivery` settings on messages now survive job

controller restarts. This addresses all of the points on the above list except the second one - use of a Deferred-delivery: header field still provides no mechanism for informing the client whether or not the setting will be honored.

However, as a purely practical matter, the mechanism chosen to provide delayed release of messages is likely to be dictated by the choice of email client and what mechanisms it supports.

33.3.22.8 Header-based message expiration(`expirysource`, `expirysource`)

(New in Messaging Server 7.0.) The `expirysource` channel option instructs the MTA to honor Expiry-date: header fields - messages will be returned as undeliverable if the time specified by this header field is exceeded. `noexpirysource` disables this check and is the default.

33.3.22.9 Maximum number of simultaneous jobs for this channel (`maxjobs`)

The `maxjobs` channel option places an upper bound on the total number of service jobs that can be created. This option must be given an integer argument; 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 [Job Controller pool](#) the channel uses.

See also the `job_limit` Job Controller option.

Note that a `imsimta cache -change -channel=NAME -job_limit=N` command can change the effective `maxjobs` value for a channel "on the fly".

33.3.22.10 Priority of messages to be handled by periodic jobs (`minperiodicnonurgent`, `minperiodicnormal`, `minperiodicurgent`, `maxperiodicnonurgent`, `maxperiodicnormal`, `maxperiodicurgent`)

OBSOLETE: These channel options have no effect nowadays, with the Job Controller. See instead [Job Controller priority-based processing](#).

In the past: When periodic delivery jobs were used, they normally processed all messages queued for the channel. However, on some channels it may be desirable to limit normal periodic job processing to only messages of specified priorities. Other special site-supplied periodic jobs may then process the remaining messages. For instance, a site might choose to have normal MTA periodic jobs pass over nonurgent messages, leaving those nonurgent messages to be delivered by some site-supplied job (perhaps scheduled to run at off-peak hours).

The `minperiodicnonurgent`, `minperiodicnormal`, or `minperiodicurgent` channel options specify the minimum priority of message that a periodic job should try to deliver; the job will ignore messages of lower priority. The `maxperiodicnonurgent`, `maxperiodicnormal`, or `maxperiodicurgent` options specify the maximum priority

of message that a periodic job should try to deliver; the job will ignore messages of higher priority.

33.3.22.11 Per-channel MT-PRIORITY control (`mtprioritiesallowed`, `mtprioritiesrequired`)

`mtprioritiesallowed` and `mtprioritiesrequired` are new in the 8.0 release. These channel options enable the MTA's support of [RFC 6710 \(SMTP Extension for Message Transfer Priorities\)](#).

The `mtprioritiesallowed` source channel option specifies the range of MT-PRIORITY values that will be accepted. MT-PRIORITY values outside this range will be adjusted up or down so they fall within the allowed range. If a single argument is given, it specifies the highest priority value that will be accepted. The default if this option is not specified is for the MT-PRIORITY extension not to be offered and for MT-PRIORITY parameters not to be accepted.

The `mtprioritiesrequired` source channel option specifies the range of MT-PRIORITY that will be accepted for enqueue. If a single argument is given, it specifies the lowest priority value that will be accepted. The message will be rejected if the message's specified MT-PRIORITY value, or if the default MT-PRIORITY value of 0 (assumed if MT-PRIORITY was not specified in the SMTP transaction), falls outside the required range with the SMTP error:

```
550 5.7.0 Message priority outside curretly allowed range
```

With either channel option, two integer arguments specify the range. Each argument must be an integer in the range -9..9. The arguments can be given in any order.

33.3.22.12 Service job pool usage (`pool`)

The Messaging Server's [Job Controller](#) creates channel jobs (jobs to deliver messages) as needed: the MTA's enqueueing processes inform the Job Controller regarding newly enqueued messages, and the [Job Controller then decides](#) which existing service job whould attempt the message's delivery, or creates a new service job, as needed.

To manage the allocation of channel delivery job processes, the Job Controller has "[processing pools](#)" (in old PMDF terminology, "queues"). Different channels may be configured to run in different processing pools via the `pool` channel option. (The old PMDF channel keyword `queue` is essentially synonymous.)

Note that for iMS/JES MS/the Oracle Messaging Server, the priority-sensitive queues of PMDF days are obsolete. Instead, the [Job Controller takes care of managing different priority messages](#) in different priority internal processing "queues" within a processing pool. That is, priority-sensitive sorting of messages is handled internally by the Job Controller, without needing explicit configuration.

33.3.22.13 Triggering new jobs (`threaddepth`)

The `threaddepth` channel option tells the [Job Controller](#) when to start a new channel "job" to handle messages: for multithreaded channels, when to start a new thread (if the process is allowed to have more threads) or failing that a new process (if more processes are allowed); for single threaded channels, when to start a new process (if more processes are allowed).

For multithreaded channels, the `threaddepth` channel option controls how many messages are handled in any one thread before the channel will consider starting to use another thread.

In particular, the MTA's `SMTP client` (for channels not marked with the `daemon` channel option) sorts outgoing messages to different destinations to different threads. The `threaddepth` channel option may be used to instruct the MTA's 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 value specified must be greater than 1 and less than 10000. The default as of JES MS 6.0 is `threaddepth 10`. (This is a change from previous versions, in which the default was 128.)

Use of `threaddepth` may be of particular interest for achieving multithreading with `daemon router` on a `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.

Similarly, the `threaddepth` option affects operation of the multithreaded `ims-ms channel`.

For single threaded channels, such as the `conversion, process, and reprocess channels`, the `threaddepth` channel option controls how many messages are handled in a single process; more messages cause the `Job Controller` to create another process (up to the `maxjobs` channel option setting for the channel and the `job_limit` Job Controller option value for the `pool` in which the channel runs) to process the messages.

33.3.22.14 user Option Under channel

The `user` channel option is used on `pipe channels` to indicate under what Unix user id to run.

In the 8.0 release and later, this option is deprecated and the `pipeuser` option from `restricted.cnf` is used instead.

33.3.23 Sensitivity limits channel options

Several channel options set channel-specific message sensitivity limits.

33.3.23.1 Sensitivity checking (`sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate`, `sensitivitycompanyconfidential`)

The `sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate`, and `sensitivitycompanyconfidential` channel options set an upper limit on the sensitivity of messages that may be accepted by a destination channel. The default is `sensitivitycompanyconfidential`; *i.e.*, messages of any sensitivity are allowed through. A message with no Sensitivity: header is considered to be of normal, *i.e.*, lowest, sensitivity. Messages with a higher sensitivity than that specified by such a channel option will be rejected when enqueued to the channel with an error:

```
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.

33.3.24 Sieve filters and delivery flags channel options

There are a number of channel options controlling [Sieve filters](#) and message envelope and delivery flags.

33.3.24.1 Address type flags (`addrtypescan`, `addrtypescanbccdefault`, `noaddrtypescan`)

(New in 7.0.5.) If the `addrtypescan` channel option is set, then RCPT TO addresses (that is, envelope recipients) are compared with header recipient fields (To:, Cc:, Bcc:, Resent-To:, Resent-Cc:, and Resent-Bcc:). When a match is found, that fact is recorded in the delivery flags associated with that envelope recipient. Those flags are then used when generating the [report part](#) of Microsoft[®] Exchange 2007 [envelope journaling](#) archive messages, distinguishing between various types of envelope recipient addresses.

`addrtypescanbccdefault` operates in the same way as `addrtypescan`, except that when no matches are found for a given address, that address is assumed to be a blind carbon (Bcc:) recipient. This option should only be used when it is certain the messages have come directly from a client that implements Bcc: by simply omitting the blind carbon recipient from the header and which doesn't support any form of local mailing lists. Use in any other context is *guaranteed* to result in incorrect types being attached!

`noaddrtypescan` is the default.

Also note that since the MTA's delivery flags are used to store this information, the MTA's delivery flag transfer facilities may be used to transport this information between MTAs; see the [deliveryflags channel option](#).

33.3.24.2 Delivery flags (`deliveryflags`, `flagtransfer`, `noflagtransfer`)

The `deliveryflags` channel option may be placed on source or destination channels. It takes a required, bit-encoded integer argument, which controls various options regarding message delivery:

Table 33.12 `deliveryflags` MTA option bit values

Bit	Value	Usage
0	1	Interpret subaddresses as folder names for delivery
1	2	When placed on a source channel, enable quota bypass (that is, delivery even if the recipient user is overquota) for messages enqueued by this channel
2	4	Messages do not require SMTP dot stuffing
4	16	Force single copy per recipient
5	32	Ignore discard or jettison actions (e.g., from Sieve filters). This corresponds to setting the message envelope bit normally set automatically by the MTA when applying a <code>discard</code> or <code>jettison</code> action, enqueueing to the <code>FILTER_DISCARD</code> or <code>FILTER_JETTISON</code> channel, so that if a "retrieval" procedure should later be performed on such a "discarded" message (such as moving the message to the reprocess channel), the message

		will then get delivered bypassing any discard or jettison actions.
6	64	When placed on a destination channel, enable transfer of delivery flags; equivalent to the <code>flagtransfer</code> channel option
7	128	Messages enqueued to the channel are considered, for purposes of CONVERSIONS mapping table testing, to have a conversion tag set
8	256	Handle as if address is a result of a "redirect" action
9	512	Handle as if SMTP AUTH (SASL) had been used as far as access checks are concerned
10	1024	Handle as if TLS had been used
11	2048	Handle as if address produced by alias

The default value for `deliveryflags` is 0.

The `flagtransfer` channel option may be placed on a SMTP server or LMTP server channel. It causes the server to advertise support of the XDFLG and XAFLG private SMTP/LMTP extension parameters to the RCPT TO command. If a Messaging Server SMTP client is sending to an SMTP server (or a Messaging Server LMTP client is sending to an LMTP server) that supports this extension, then that SMTP client (or LMTP client) will pass along (transfer) the delivery flags. For instance, this can be useful when user filters (performing `fileinto` Sieve operations) will be performed on a "front-line" system that must then relay the messages to a "back-end" system. `noflagtransfer` disables delivery flag transfer and is the default.

Setting `flagtransfer` is equivalent to setting bit 6 (value 64) of the `deliveryflags` channel option.

33.3.24.3 Filter file location (`filter`, `nofilter`, `destinationfilter`, `nodestinationfilter`, `sourcefilter`, `nosourcefilter`, `disablesourcefilter`, `disabledestinationfilter`)

For the Messaging Server MTA, user and group [Sieve filters](#) are normally enabled simply by storing them in the users' (and groups') `mailSieveRuleSource` attribute - or more precisely, storing them in the attribute named by the `ldap_filter` MTA option. However, user filters may be located in an alternate sort of location (in files on disk, for instance) via use of the `filter` channel option. So while not normally used, in principle the `filter` option may be used on the `ims-ms`, `native`, and `tcp_lmtpc*` 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; it means that only filters enabled implicitly via user/group LDAP entries will be used (and no additional, external Sieve lookup will be performed by the MTA).

The `sourcefilter` and `destinationfilter` channel options may be used on general MTA channels to specify a channel-level filter to apply to incoming and outgoing messages, respectively. (More precisely, a `sourcefilter` is applied when the source channel on which it is specified is enqueueing a message; a `destinationfilter` is applied when any channel is enqueueing a message to the destination channel on which it is specified.) These channel options take a required URL argument describing the channel filter file location. `nosourcefilter` and `nodestinationfilter` are the defaults and mean that no channel mailbox filter is enabled for either direction of the channel.

The obsolete `channefilter` and `nochannefilter` keywords are synonyms for `destinationfilter` and `nodeestinationfilter`, respectively.

New in Messaging Server 7.0 update 3 are the `disablesourcefilter` and `disabledestinationfilter` keywords. These keywords can be used to suppress the evaluation and interpretation of Sieve filters based on source or destination channel, respectively. Each keyword takes a single nonnegative integer argument, whose value is interpreted as follows:

Table 33.13 `disablesourcespamfilter` and `disabledestinationpamfilter` MTA options values

Value	Usage
0	Disable all Sieves
1	Only spam filter Sieves are evaluated and interpreted
2	Only spam filter and source channel Sieves are evaluated and interpreted
3	Spam filter and source channel Sieves, and the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), are evaluated and interpreted
4	Spam filter, source channel, the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), and destination channel Sieves are evaluated and interpreted
5	Spam filter, source channel, the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), destination channel, and <code>ORIG_SEND_ACCESS</code> Sieves are evaluated and interpreted
6	Spam filter, source channel, the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), destination channel, <code>ORIG_SEND_ACCESS</code> , and <code>SEND_ACCESS</code> Sieves are evaluated and interpreted
7	Spam filter, source channel, the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), destination channel, <code>ORIG_SEND_ACCESS</code> , <code>SEND_ACCESS</code> , and <code>ORIG_MAIL_ACCESS</code> Sieves are evaluated and interpreted
8	Spam filter, source channel, the <code>systemfilter</code> MTA system Sieve (which in legacy configuration was <code>imta.filter</code>), destination channel, <code>ORIG_SEND_ACCESS</code> , <code>SEND_ACCESS</code> , <code>ORIG_MAIL_ACCESS</code> and <code>MAIL_ACCESS</code> Sieves are evaluated and interpreted

33.3.24.4 Sieve filter `fileinto` action channel options (`fileinto`, `nofileinto`)

The `fileinto` channel option, currently only especially meaningful for channels delivering into the Messaging Server Message Store (that is, `ims-ms` and `tcp_lmtpc*` channels), specifies how to alter an address when a Sieve filter "`fileinto`" action is applied. `nofileinto` is the default, and means that a Sieve filter "`fileinto`" action has no address-modifying meaning for that destination channel.

For `ims-ms` channels, the usual usage is

```
fileinto $U+$S@$D
```

meaning that the folder name should be inserted as a [subaddress](#) into the original address, replacing any originally present subaddress. (The default value for the [FILEINTO ims-ms-channel-specific option](#) then results in the `ims-ms` channel interpreting that subaddress as a request for folder delivery.)

For [tcp_lmtpcs* channels](#), the usual usage is

```
fileinto @$4O:$U+$S@$D
```

(where note that in `$4O` the `O` is the capital or majuscule letter "o", *not* the numeral zero 0). The effect is that the explicit source route to the mailhost should be preserved if present, and the foldername should be inserted as a [subaddress](#) into the original address, replacing any originally present subaddress.

The Message Store delivery code normally considers any "trusted" [subaddress](#) present on a recipient address as a request to deliver directly into the correspondingly named folder. (This can be overridden for the `ims-ms` channel by disabling the [FILEINTO ims-ms-channel-specific option](#).) Application of the `fileinto` channel option also sets a [bit in the message envelope](#) that means that for Message Store delivery "trust this subaddress as a folder name for delivery purposes". So when the `fileinto` channel option is applied on an [ims-ms channel](#) or a [tcp_lmtpcs* channel](#), subaddresses added due to a Sieve filter "fileinto" action will cause folder delivery. Note that unless a subaddress has been added/replaced due to such a Sieve "fileinto" action and the `fileinto` channel option's resulting setting of the proper message envelope bit, any other subaddress will not normally be considered as a valid request for folder delivery---not unless the IMAP post ACL ([RFC 4314, IMAP4 Access Control List \(ACL\) Extension](#)) has been set on that folder in the Message Store.

33.3.24.5 Sieve filter and delivery flags channel options: `scriptlimit (integer)`, `sievelimit (integer)`, `sizelimit (integer)`

The `scriptlimit`, `sievelimit`, and `sizelimit` source channel options set limits on how many Sieve scripts a user may have, how large each individual script may be, and the total size limit with scripts combined.

33.3.25 Size limits on messages channel options

A number of channel options relate to message size limits. See also the [Message size MTA options](#). The SMTP server also has some message size related limits; see [TCPIP-channel-specific options](#). And for web-based email clients, see also the `maxmessagesize` MSHTTP option.

33.3.25.1 Triggering alternate channel processing (`alternatchannel`, `alternateblocklimit`, `alternatelinelimit`, `alternaterecipientlimit`)

It is sometimes useful to force processing of messages meeting certain criteria to occur on a channel distinct from the one chosen by the MTA's alias expansion and rewriting process. The `alternatchannel` channel option provides a means to specify such a channel while the `alternateblocklimit`, `alternatelinelimit`, and `alternaterecipientlimit` channel options specify the criteria for when the alternate channel should be used.

`alternatechannel` takes the name of the alternate channel to use as an argument. `alternateblocklimit` takes an unsigned integer as an argument; the alternate channel will be used if the computed block size of the message exceeds this value. `alternaterecipientlimit` also takes an unsigned integer argument; the message will be queued to the alternate channel if the number of recipients queued to the current channel exceeds this value. Finally, the `alternatelinelimit` channel option takes an unsigned integer argument; the alternate channel will be used if the computed number of lines in the message exceeds this value.

Note that `alternaterecipientlimit` is a limit on envelope recipients for this message copy, on this channel; it has nothing to do with how many addresses may or may not be in the header; and envelope recipients on other channels are also irrelevant. However, the `alternaterecipientlimit` check does get performed before any message copy split-up due to storage of recipients per file controls such as `addrsperfile`, `single`, or `single_sys` channel option application.

Note that any `*SEND_ACCESS` or `*MAIL_ACCESS` mapping table probes will use the "original" destination channel name, not the alternate destination channel name, but a `CONVERSIONS` mapping table probe will use the alternate destination channel name.

Note that the alternate channel selection process is neither iterative nor recursive: Once an alternate channel has been selected it will be used regardless of what the various alternate channel options on that channel say to do.

33.3.25.2 Message size limits (`blocklimit`, `linelimit`, `sourceblocklimit`, `noblocklimit`, `nolinelimit`)

Although fragmentation may be used to break messages into smaller pieces automatically, it may also be appropriate in some cases to simply reject outright messages larger than some administratively defined limit, (e.g., so as to avoid service denial attacks on the system or individual mailboxes).

The `blocklimit`, `linelimit` and `sourceblocklimit` channel options are used to impose absolute size limits. Each of these options requires a single integer argument. `blocklimit` specifies the maximum number of blocks allowed in a message. The MTA will reject attempts to queue messages containing more blocks than this to the channel. The `sourceblocklimit` specifies the maximum number of blocks allowed in an incoming message. The MTA will reject 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` MTA option. `linelimit` specifies the maximum number of lines allowed in a message. The MTA will reject attempts to queue messages containing more than this number of lines to the channel. These limits can be imposed simultaneously if necessary.

Note that the `line_limit` and `block_limit` MTA options can be used to impose similar limits on all channels. These limits have the advantage that since they apply across all channels the MTA can make them known to mail clients via the SMTP SIZE extension prior to obtaining message recipient information. This simplifies the process of message rejection in some situations.

The `nolinelimit` and `noblocklimit` settings are the default and mean that no per-channel limits are imposed. But message size may still be limited via other configuration choices: global limits imposed via the `line_limit` or `block_limit` MTA options, per-

domain limits imposed via the attributes named by the `ldap_domain_attr_blocklimit` or `ldap_domain_attr_sourceblocklimit` MTA options, or per-user limits imposed via the attributes named by the `ldap_blocklimit` or `ldap_sourceblocklimit` MTA options, or a per-group/mailling list limit imposed via the attribute named by the `ldap_maximum_message_size` MTA option, or per-group/mailling list limits imposed via the `alias_blocklimit` or `alias_linelimit` alias options (or in legacy configuration the equivalent [BLOCKLIMIT] or [LINELIMIT] [alias file named parameters](#)), or a limit imposed via use of the `$$S` flag in the [FROM_ACCESS mapping table](#).

Note that the `block_limit` MTA option, or similarly the `sourceblocklimit` channel option on an incoming TCP/IP channel, causes the MTA's SMTP server to advertise the stated size (the minimum of `block_limit` and the channel's `sourceblocklimit`) as the maximum size accepted in response to a sender's EHLO command. This means that clients/senders that understand the SIZE SMTP extension (see [RFC 1870](#), ESMTP message size extension) may not bother to even *try* to submit larger messages after seeing the MTA's EHLO response. Or if they do start to submit a larger message, if they at least specify the SIZE via the SMTP extension on the MAIL FROM command, then the MTA's SMTP server will reject the message at the MAIL FROM. This contrasts with the effect of the `blocklimit` channel option which cannot be applied (even if the sender used the SIZE extension on the MAIL FROM) until after the RCPT TO is specified so that the MTA can determine the `blocklimit` for the relevant destination channel.

Related to the above effects, is the fact that messages rejected due to the `block_limit` MTA option, or similarly due to the `sourceblocklimit` channel option, do not necessarily result in a "J" rejection entry in the `mail.log*` file, since potentially a client/sender-SMTP doesn't even bother to submit a message when it sees the advertised size limit. Or even if the MTA does perform a "rejection" itself, it may occur at the MAIL FROM stage (if the client uses the SIZE SMTP extension to include the expected message size on the MAIL FROM command line); this is before the MTA has its normal log information (such as recipient information), but will nevertheless be logged as a "J" record (missing some fields such as recipient fields) as of JES MS 6.0 and later. (In earlier versions, such rejections at the MAIL FROM stage would not be recorded in a `mail.log*` record; in particular, would not cause a "J" record to be generated.) This contrasts with messages rejected due to `blocklimit` on the destination channel which do, and have even in older versions, get logged as "J" entries with recipient field(s) filled in "normally".

33.3.25.3 Expansion of multiple addresses (`expandlimit`, `expandchannel`, `holdlimit`)

Most MTA 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 ("on-line" delays). In particular, multi-recipient messages that require a great deal of processing of the message body can be affected by processing delays, or that require creation of many different file copies on disk in the MTA queue area can be affected by slow disk performance. If the resulting delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

The MTA provides a special facility to force deferred ("off-line") processing of additional recipient message copies once a given number of addresses are specified for a single message. The deferral happens after processing of the "initial" recipients (those before the `expandlimit` value was reached), and after address processing for the additional recipients too, (for instance, after [*_ACCESS mapping table](#) checks and after alias processing), but before

message processing. In particular, such deferral means that for the "additional" recipients, only *one* message file (storing all of the "additional" recipients), is written to the queue area (to a [reprocess*](#) or [process*](#) channel queue area, depending upon use of the `expandchannel` channel option). 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` channel option. The `expandlimit` option takes an integer argument that specifies at what number of recipients to begin deferring processing of the message copy (or copies) to that and additional recipient addresses. The default value is effectively infinite if the `expandlimit` channel option is not specified. A value of 1 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` channel option 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` channel option; 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 some purposes. In particular, for Sun ONE Messaging Server MTA versions 5.2 and earlier, typical configuration usage required that a processing channel, rather than a reprocessing channel, be used. 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` option to have any effect. If your configuration was built by the initial configuration utility, then you should already have a reprocessing channel.

(Note that typical Messaging Server sites running version 5.2 or earlier could not use the `expandlimit` option unless they also marked the affected channel `expandchannel process` (or `expandchannel process_somethingorother` redirecting the expansion to an alternate `process_*` sort of channel), as enqueues to a channel marked `viaaliasrequired` would not succeed if deferred to a `reprocess*` channel.)

Extraordinarily large lists of recipient addresses are often a characteristic of so-called SPAM---junk e-mail. The `holdlimit` channel option 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` channel option). As `.HELD` messages, the files will sit unprocessed in that MTA queue area awaiting manual intervention by the MTA postmaster.

33.3.25.4 Message size affecting priority (`urgentblocklimit`, `normalblocklimit`, `nonurgentblocklimit`, `secondclassblocklimit`)

Note that as of the 8.0 release, these size-based priority override channel option effects are nullified if the MT-PRIORITY SMTP extension has been used to set an explicit priority value.

The `urgentblocklimit`, `normalblocklimit`, `nonurgentblocklimit` and `secondclassblocklimit` channel options may be used to instruct the MTA to downgrade the priority of messages based on size. These options all require a single integer argument specifying the message size, in MTA blocks, at which to perform the priority downgrading. An

MTA block is normally 1024 bytes; this can be changed with the `block_size` MTA option. This effective priority, in turn, affects the Job Controller's scheduling of delivery attempts, higher priority messages normally being attempted before lower priority messages, or see the `*_delivery` Job Controller options for further control over the scheduling of even initial message delivery attempts, or the `*backoff` channel options for further control over the scheduling of additional delivery attempts, as well as the `*notices` channel options for further control over the "timing out" (bouncing) of undelivered messages.

The `urgentblocklimit` channel option instructs the MTA to downgrade messages larger than the specified size to normal priority. The `normalblocklimit` channel option instructs the MTA to downgrade messages larger than the specified size to nonurgent priority. The `nonurgentblocklimit` channel option instructs the MTA to downgrade messages larger than the specified size to second-class priority. Finally, the `secondclassblocklimit` instructs the MTA to downgrade messages larger than the specified size to third-class priority.

Note: Both second-class and third-class are nonstandard priority values.

33.3.26 Spamfilter channel options

The MTA supports use of up to eight external spam/virus filter packages. For each such spam/virus filter package, there is a set of channel options selecting which channel(s) invoke the spam/virus filter package. For conciseness, listed here are solely the options for spam/virus filter package 1; but note that there are analogous channel options `*spamfilterN*` for $N=2, \dots, 8$ as well.

See also the [Spamfilter MTA options](#) for configuration of the location and operation of the spam/virus filter package interfaces.

As of Messaging Server 7.0.5, [imexpire also supports invoking spam/virus filter packages](#); in particular, this can be used to perform post-delivery removal of spam/virus-infected messages from the Message Store. For this purpose, `imexpire` is told a source channel "as which" to execute, and then any `sourcespamfilterN` or `sourcespamfilterNoptin` for that channel will be invoked by `imexpire`.

33.3.26.1 `destinationspamfilterN`, `destinationspamfilterNoptin`, `sourcespamfilterN`, `sourcespamfilterNoptin`, `disabledestinationspamfilterN`, `disablesourcespamfilterN` Channel Options

The `destinationspamfilterN`, `destinationspamfilterNoptin`, `sourcespamfilterN`, `sourcespamfilterNoptin`, `disabledestinationspamfilterN`, and `disablesourcespamfilterN` [channel options](#) control whether spam/virus filter package processing is invoked by the MTA during message enqueue processing.

As of Messaging Server 7.0.5, the [imexpire utility is also capable of invoking spam/virus filter packages](#), "as if" it were an MTA channel. For such use, `imexpire` is told a source channel as which to operate, and then any `sourcespamfilterN` and `sourcespamfilterNoptin` channel options for that channel specify what spam/virus filter package(s) to invoke.

The MTA supports the use of up to eight distinct spam/virus filtering packages, as configured via the `spamfilterN_*` MTA options (with N ranging between 1 and 8); the set of options

with the same number all configure one of the distinct packages. And the number in a `*spamfilterN*` channel option correlates with which spam/virus filter package is to be invoked (or not invoked). (Note that the `*brightmail*` options, and the `*spamfilter*` options without an explicit number, are deprecated synonyms for the `*spamfilter1*` options. These deprecated forms are not allowed in a Unified Configuration.)

The `*spamfilterNoptin` channel options not only trigger spam/virus filter package processing, but do so with a particular "optin" value or values set, as some spam/virus filter packages (such as Brightmail) support different "choices" or "optin" values for what type of filtering will be performed (e.g., spam vs. virus). The MTA supports use of multiple, comma-separated "optin" values as an argument to an `*optin` channel option. Spam/virus filter package processing may also be triggered, with a specific "optin" value or values, via use of a per-user attribute (see the [ldap_optinN MTA options](#)) or a per-domain attribute (see the [ldap_domain_attr_optinN MTA options](#)). When "optin" values from multiple sources apply for a message -- for instance, if an "optin" value is set via a channel option as well as via an attribute---the MTA will pass all the applicable "optin" values to the spam/virus filter package.

The `disable*spamfilterN` channel options can be used to disable, on a channel-specific basis, spam/virus filtering that would otherwise be performed. In particular, note that the `disable*spamfilterN` channel options override any user-level optin to spam/filter package processing (as for instance selected via an LDAP attribute named by an [ldap_optinN MTA option](#)), and override any channel level optin (as for instance via a `*spamfilternoptin` channel option). For instance, one might use `disableourcespamfilter1` on a [tcp_auth channel](#) to disable spam/virus filtering (by spam/virus filter package number 1) for all messages coming in the `tcp_auth` channel, overriding any spam/virus filter package use that might normally be triggered due to the destination channel or recipient address(es).

33.3.27 SMTP and LMTP protocol channel options

For additional channel options affecting the SMTP protocol, specifically those relating to the SMTP extensions AUTH or STARTTLS (SASL or TLS use), see [TLS and SASL channel options](#).

For additional channel options affecting TCP/IP connections and DNS lookups, see [TCPIP connections and DNS lookups channel options](#).

33.3.27.1 Receiving an SMTP ETRN command (`allowetrn`, `blocketrn`, `disableetrn`, `domainetrn`, `silentetrn`)

The `allowetrn`, `blocketrn`, `disableetrn`, `domainetrn`, and `silentetrn` channel options affect the MTA's response when a sending SMTP client issues the SMTP ETRN command, requesting that the MTA attempt to deliver messages in the MTA's queues. See [RFC 1985](#) for the specification of the SMTP ETRN command syntax. In particular, note that the MTA's SMTP server interprets a received `ETRN hostname` command as a request to deliver all messages for `hostname`, a received `ETRN #channelname` command as a request to run the `channelname` channel, and a received `ETRN @hostname` command as a request to run the channel which `hostname` rewrites to.

`allowetrn` means that the MTA will attempt to honor all ETRN commands and will echo back the name of the channel that will be run in response to the ETRN command. `silentetrn` tells the MTA to honor all ETRN commands, but without echoing back the name of the channel which the domain matched and which the MTA will hence be attempting to run. `domainetrn` tells the MTA to honor only those ETRN commands that specify a domain;

it also causes the MTA not to echo back the name of the channel which the domain matched and which the MTA will hence be attempting to run. `disableetrn` disables support for the ETRN command entirely; ETRN will not be advertised by the SMTP server as a supported command. The default behavior, if none of these channel options is explicitly specified, corresponds most closely to `silentetrn`.

When ETRN is permitted (`allowetrn`, `domainetrn`, or `silentetrn` is set, or no option is set), the [ETRN_ACCESS mapping table](#) can be used to exert more precise control over which SMTP clients are allowed to issue which ETRN commands (and optionally control over what channel is actually run as a result of the ETRN command).

The `blocketrn` channel option tells the MTA not to honor an ETRN command if the ETRN command attempts to run that channel. Note that this channel option is therefore relevant on a destination channel, not on the incoming TCP/IP channel (unless that incoming channel would also be the destination channel for an attempted ETRN command). Note that having `disableetrn` set on a destination channel also has this effect.

Also see the discussion of the [ALLOW_ETRNS_PER_SESSION](#) SMTP channel setting, which may be used to limit the number of ETRN commands which the MTA will honor during a single session.

33.3.27.1.1 ETRN_ACCESS mapping table

When the MTA's SMTP server is configured to support (at least some uses of) the ETRN command ([allowetrn](#), [domainetrn](#), or [silentetrn](#) is set, or the default behavior when no `*etrn` option is set), then the ETRN_ACCESS mapping table can be used to exert more precise control over which SMTP clients are allowed to issue which ETRN commands (and optionally control over what channel is actually run as a result of the ETRN command). Probes of the ETRN_ACCESS mapping table have the form:

```
transport-info | app-info | channel-to-run | full-name | claimed-system
```

(Here `claimed-system` is the ETRN parameter, and `full-name` is a processed version of that parameter. See discussion of the [PORT_ACCESS mapping table](#), or the [MAIL_ACCESS mapping table](#), for discussion of the `transport-info` and `app-info` portions of the probe string.) If the mapping table returns a `$N`, `$n`, `$F`, or `$f`, the ETRN command is rejected with a "459 4.5.0" error. If the mapping table returns a `$S` or `$s`, the ETRN is attempted. If the mapping table also returns a `$Dchannel-name` or `$dchannel-name`, then the MTA tries to lookup `channel-name` (in the channel/host table from the configuration file) and if that lookup is successful, runs that channel (rather than whatever channel the original ETRN command might have run).

33.3.27.2 Binary SMTP (`binaryclient`, `nobinaryclient`, `binaryserver`, `nobinaryserver`)

The BINARYMIME SMTP extension defined in [RFC 3030](#) provides support for transferring messages containing binary parts without encoding over SMTP.

New in MS 8.0, the `binaryserver` source channel option enables this extension in the SMTP server. The `nobinaryserver` source channel option disables it, and is the default. Note that `binaryserver` enables the BDAT command for BODY=BINARYMIME messages even if [chunkingserver](#) is not in effect.

Binary messages submitted using this extension are immediately converted by the MTA to regular 8bit MIME so the format of messages in the MTA queues is not affected, nor is the format of messages that are passed through the spam filter interface.

The `BINARYMIME` extension is not presently supported by the SMTP client, so the `binaryclient` and `nobinaryclient` channel options are currently unimplemented.

33.3.27.3 SMTP EHLO command (`ehlo`, `checkehlo`, `noehlo`, `refuseehlo`)

The SMTP protocol has been extended ([RFC 1869](#)) to allow for negotiation of additional commands. This is done via 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 in-compliant servers are known to drop the connection on receipt of an unknown command. And in some cases use of facilities negotiated by EHLO will confuse a standards-in-compliant proxy intermediary.

The MTA's [SMTP client](#) implements a strategy whereby it will attempt to reconnect and use HELO when any server drops the connection on receipt of an EHLO. However, this strategy still may not work if the remote server not only drops the connection but also goes into a problematic state upon receipt of EHLO.

The channel options `ehlo`, `noehlo`, and `checkehlo` are provided to deal with such situations. `ehlo` tells the MTA to use the EHLO command on all initial connection attempts. `noehlo` disables all use of the EHLO command. `checkehlo` tests the greeting 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 option corresponding to this default behavior, which lies between the behaviors resulting from the `ehlo` and `checkehlo` options.

Finally, in the event of the MTA's SMTP server not working properly with a remote SMTP client due to the use of some SMTP extension that isn't understood by a broken intermediate proxy, the `refuseehlo` channel option can be used to simultaneously disable client use of EHLO and cause the SMTP server to refuse to accept EHLO, insisting on HELO instead. Note that this option should be used sparingly if at all: A far superior approach is to either fix or eliminate the in-compliant intermediary.

33.3.27.4 Recipient validity date check (`checkrrvs`, `ignorerrvs`)

The `checkrrvs` and `ignorerrvs` source channel options, for support of [RFC 7293](#), are new in 8.0. `ignorerrvs` is the default.

`ignorerrvs` means that the SMTP server will not offer or support the RRVS SMTP extension. In particular, client attempts to use an RRVS parameter in a RCPT TO command will cause an error, and the MTA will ignore any Require-Recipient-Valid-Since: header line.

`checkrrvs` when set on an SMTP source channel means that the SMTP server offers and supports use of the SMTP RRVS extension. In particular, the MTA will check for a valid date for recipient mailbox ownership, whether specified (preferentially) in a RRVS parameter in the RCPT TO command, or in a Require-Recipient-Valid-Since: header field, and check for a valid date for the domain in the recipient address.

If the `checkrrvs` check fails on the recipient mailbox address, the recipient will be rejected with the SMTP error:

```
550 5.7.15 account information on file is older than actual user account
```

or alternate text as controlled by the `error_text_wrong_account` MTA option; if the `checkrrvs` check fails due to the domain creation date, the recipient will be rejected with the SMTP error:

```
550 5.7.18 domain owner has changed
```

or alternate text as controlled by the `error_text_wrong_domain` MTA option.

33.3.27.5 Chunking SMTP (`chunkingclient`, `nochunkingclient`, `chunkingserver`, `nochunkingserver`)

[RFC 3030](#) defines the CHUNKING extension to SMTP. Chunking provides an alternative BDAT command that can be used instead of the normal DATA command to transfer message content. BDAT uses octet counts rather than dot-stuffing and hence is more efficient. The `chunkingclient` option, the default, tells the SMTP client to use BDAT if the server says it supports it. `nochunkingclient` disables this usage.

The `chunkingserver` option tells the SMTP server to announce support for and allow use of the CHUNKING extension. `nochunkingserver` disables chunking support in the server. `chunkingserver` is the default.

33.3.27.6 Channel operation type (`submit`, `relay`, `passthrough`, `conditionalpassthrough`, `conditionalrelay`)

The MTA supports the concept of a general "operating mode" for message enqueue. Different modes provide different levels of message inspection, fixup, and error checking. There are four basic modes: default, `submit`, `relay`, and `passthrough`.

The MTA supports [RFC 6409](#)'s Message Submission protocol (which is an update of [RFC 4409](#), which is in turn an update of [RFC 2476](#)). The `submit` source channel option 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](#), since updated by [RFC 6409](#), established port 587 for such [message submission](#) use.

Note that a channel marked `submit` has ETRN unconditionally disabled; that is, it gets the effect of `disableetrn`, irrespective of any `*etrn` channel option setting. A channel marked `submit` will also add a Date: header line, if one was missing from the original submitted message, *without* also adding a Date-Warning: header line; that is, since submission of messages without the (normally required for regular SMTP submission) Date: header line is legal on the Message Submission port, the MTA does not flag such messages originally lacking a Date: header line with the Date-Warning: header line it would generate in the case of such messages improperly submitted to the regular SMTP port.

Proper practice (configuration) on a `submit` channel includes requiring some form of user authentication, typically use of SMTP AUTH, and permitting (if not requiring) use

of STARTTLS; see [RFC 6409 \(Message Submission for Mail\)](#). Thus a properly configured `submit` channel should normally be marked with `mustsaslseserver` and `maytlserver` (or `musttlserver` if a site wishes to require STARTTLS use). Although requiring use of SMTP AUTH is what Message Submission channels *should* do, sites that wish to allow message submission without authentication (submission to the Message Submission port without requiring SMTP AUTH) *may* do so *if* enforcing some other form of sender verification, such as limiting such submissions only to certain "trusted" IP sources; see for instance the [FROM_ACCESS mapping table](#) which may be used to enforce IP source based restrictions.

Relay mode performs fewer checks than submit mode, however, certain structural message problems will cause message enqueue to fail with an error. For example, a missing `Date:` header field will result in an SMTP-level error in relay mode.

Relay mode is specified by adding the `relay` option to the appropriate source channel.

Passthrough mode disables as many checks and message modification steps as practical. For example, a message that is missing its required `Date:` header field will not have one added.

Passthrough mode is specified by adding the `passthrough` option to the appropriate source channel.

IMPORTANT NOTE: Passthrough is an EXTREMELY DANGEROUS setting because it disables a number of checks clients are known to depend on. Additionally, various internal MTA functionality that depends on message inspection taking place, such as header field canonicalization, is disabled in passthrough mode. While it is tempting to use passthrough mode in some cases to work around problems caused by agents that insist on receiving standards-incompliant messages, field experience has shown that this "cure" is almost invariably worse than the disease.

Default mode (which of course is the default) lies somewhere between submit and relays. Most available checks and fixups are performed but few will cause an enqueue failure. Continuing the example of the missing `Date:` header field, in default mode a `Date:` field will be added by the MTA and a `Date-warning:` header field will also be added explaining that this was done.

There is no option associated with default mode.

Finally, there are two additional mode setting options: `conditionalrelay` and `conditionalpassthrough`. These settings first check to see if the message already contains any `Received:` header fields. If such fields exist the mode specified in the option name is enabled, if not the MTA stays in its original mode.

For the [Sieve "environment" extension](#), the MTA supports a private item `vnd.oracle.operation-mode`, allowing Sieve scripts to discover what the current operation mode is.

New in 8.0, see the [Sieve "setoperation" action](#), which enables system-level Sieve scripts to set what mode of operation to use for a message.

33.3.27.7 Maximum allowed recipients or bad commands (`recipientlimit`, `recipientcutoff`, `deferralrejectlimit`, `disconnectrecipientlimit`, `disconnectrejectlimit`, `disconnectbadcommandlimit`, `disconnectbadburllimit`, `disconnectcommandlimit`)

The `recipientlimit`, `recipientcutoff` and (new in JES MS 6.2) `deferralrejectlimit` channel options, when placed on a source channel, impose per-channel limits on the number of recipients for a submitted message. Each of these options accepts a single integer argument; they default to no limit. (Note that setting `recipientlimit` or `recipientcutoff` to 0 has no effect; only positive limit values will be enforced.) These options are all per-channel (as opposed to per-SMTP-server) analogues of the [ALLOW_RECIPIENTS_PER_TRANSACTION](#), [REJECT_RECIPIENTS_PER_TRANSACTION](#), and [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) SMTP channel settings.

`recipientlimit` limits the total number of recipient addresses that will be accepted for the message; additional recipients will be rejected. The text in the rejection is configurable via the [error_text_recipient_over](#) MTA option, which by default is "too many recipients specified". The error is a temporary rejection by default, but if bit 4 (value 16) of the [use_permanent_error](#) MTA option is set, then the rejection is permanent. So in the case of attempted SMTP message submissions, the default temporary error, with the default error text, would appear as:

```
451 4.5.3 too many recipients specified
```

whereas with the default `error_text_recipient_over` text but with bit 4 (value 16) of the `use_permanent_error` MTA option set, then a permanent error would appear as:

```
550 4.5.3 too many recipients specified
```

`recipientcutoff` compares the total number of recipients that were presented to the MTA to the specified value and a message will not be accepted for delivery to *any* recipients if the value is exceeded. In the case of attempted SMTP submissions, the message will be rejected at the DATA command with the SMTP rejection:

```
451 4.4.5 Error ending envelope - Too many recipients specified for this message
```

New in JES MS 6.2, and supported only for SMTP (not for LMTP), `deferralrejectlimit` limits the number of bad (failing) addresses that will be allowed during a single session; after this number, all subsequent recipient addresses, good or bad, will be rejected with a temporary error. In the case of attempted SMTP submissions, additional RCPT TO: commands will be rejected with the error:

```
451 4.5.3 Too many rejections; try again later
```

while attempted VRFY: commands will be rejected with the error:

```
451 4.5.3 Verification blocked; too many rejections
```

Similar limits controlled on a per-user and per-domain basis can be configured via LDAP attributes; see the MTA options [ldap_recipientlimit](#), [ldap_recipientcutoff](#), [ldap_domain_attr_recipientlimit](#) and [ldap_domain_attr_recipientcutoff](#).

The `disconnect*` channel options are new in Messaging Server 6.2 (except: `disconnectcommandlimit` new in Messaging Server 7.1, `disconnectbadburllimit` new in Messaging Server 7.4-18.01). They are supported only for SMTP, not for LMTP. Each takes an integer argument specifying the maximum number of (recipients, rejections, bad

commands, or commands, as applicable) which will be accepted for messages submitted via that source channel; any more will result in the MTA forcing a disconnect of the SMTP session, after issuing an error response to the client consisting of (for `disconnectrecipientlimit`):

```
421 4.7.0 Session recipient limit reached; disconnecting
```

for `disconnectrejectlimit` in JES MS 6.2:

```
450 4.7.0 Session bad recipient limit reached; disconnecting
```

or in JES MS 6.3 and later (JES MS 6.3 also being when behavior was enhanced so that rejected MAIL FROM's count against the `disconnectrejectlimit`, whereas in JES MS 6.2 only rejections at the RCPT TO or VRFY stages counted; JES MS 6.3 is also when behavior was enhanced so that `disconnectrejectlimit` is checked at the VRFY stage and with a negative value applied at the VRFY stage, whereas previously such VRFY rejections were counted but the disconnect would not be triggered until a subsequent RCPT TO attempt "noticed" that the threshold was exceeded):

```
421 4.7.0 Session rejection limit reached; disconnecting
```

or (for `disconnectcommandlimit`):

```
450 4.7.0 Maximum number of commands exceeded
```

In the case of the `disconnectrecipientlimit` or `disconnectrejectlimit` channel options, once the limit is exceeded, the error-response-and-disconnect normally will occur after the next MAIL FROM or RSET command (or in the case of `disconnectrejectlimit` in JES MS 6.3 and later, potentially after a failed VRFY attempt). (Note that because the disconnect usually does not happen until after a subsequent MAIL FROM or RSET, these `disconnect*` channel options would most often be used in conjunction with other channel keywords or [TCP/IP-channel-specific option](#) settings: perhaps `recipientlimit` or `recipientcutoff` to limit the number of recipient addresses accepted, or `deferralrejectlimit` or the [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) TCP/IP-channel-specific option setting.) In the case of `disconnectbadcommandlimit`, `disconnectbadburllimit`, and `disconnectcommandlimit`, once the limit is exceeded the error response is issued and the MTA forces the disconnect.

The `disconnectbadburllimit` channel option is new in Messaging Server 7.4-18.01. A single integer parameter is accepted specifying the number of invalid [BURL commands](#) that will be allowed before disconnecting. The default is 3.

Note that VRFY attempts are counted separately from RCPT TO attempts against the recipient count; that is, one may have up to the recipient limit of VRFYs *and* up to the recipient limit of RCPT TOs. The VRFYs are counted, but counted separately from RCPT TOs. In contrast, failed VRFY attempts are added to the same rejection counter used for counting failed RCPT TO attempts and MAIL FROM attempts for purposes of comparison against the rejection limit; that is, one may have up to the rejection limit of any combination of failed VRFYs, MAIL FROMs, or RCPT TOs.

When the `deferralrejectlimit` has been reached (or a TCP/IP-channel-specific option setting of [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) has been reached), a client VRFY attempt will receive from the MTA an error response:

451 4.5.2 Verification blocked; too many rejections

Note that prior to JES MS 6.3, the error was instead:

452 4.5.2 Verification blocked; too many bad addresses.

Note that the `FROM_ACCESS` mapping table's `$$` flag may also be used to set limits such as `recipientlimit` or `recipientcutoff`.

For forcing IMAP or POP disconnection after a specified number of protocol errors -- similar to the `disconnectbadcommandlimit` effect for SMTP -- see the `maxprotocolerrors` IMAP or POP option.

33.3.27.8 Delivery flags (`deliveryflags`, `flagtransfer`, `noflagtransfer`)

The `deliveryflags` channel option may be placed on source or destination channels. It takes a required, bit-encoded integer argument, which controls various options regarding message delivery:

Table 33.14 `deliveryflags` MTA option bit values

Bit	Value	Usage
0	1	Interpret subaddresses as folder names for delivery
1	2	When placed on a source channel, enable quota bypass (that is, delivery even if the recipient user is overquota) for messages enqueued by this channel
2	4	Messages do not require SMTP dot stuffing
4	16	Force single copy per recipient
5	32	Ignore <code>discard</code> or <code>jettison</code> actions (e.g., from Sieve filters). This corresponds to setting the message envelope bit normally set automatically by the MTA when applying a <code>discard</code> or <code>jettison</code> action, enqueueing to the <code>FILTER_DISCARD</code> or <code>FILTER_JETTISON</code> channel, so that if a "retrieval" procedure should later be performed on such a "discarded" message (such as moving the message to the reprocess channel), the message will then get delivered bypassing any <code>discard</code> or <code>jettison</code> actions.
6	64	When placed on a destination channel, enable transfer of delivery flags; equivalent to the <code>flagtransfer</code> channel option
7	128	Messages enqueued to the channel are considered, for purposes of <code>CONVERSIONS</code> mapping table testing, to have a <code>conversion tag</code> set
8	256	Handle as if address is a result of a "redirect" action
9	512	Handle as if SMTP AUTH (SASL) had been used as far as access checks are concerned
10	1024	Handle as if TLS had been used

11	2048	Handle as if address produced by alias
----	------	--

The default value for `deliveryflags` is 0.

The `flagtransfer` channel option may be placed on a SMTP server or LMTP server channel. It causes the server to advertise support of the XDFLG and XAFLG private SMTP/LMTP extension parameters to the RCPT TO command. If a Messaging Server SMTP client is sending to an SMTP server (or a Messaging Server LMTP client is sending to an LMTP server) that supports this extension, then that SMTP client (or LMTP client) will pass along (transfer) the delivery flags. For instance, this can be useful when user filters (performing `fileinto` Sieve operations) will be performed on a "front-line" system that must then relay the messages to a "back-end" system. `noflagtransfer` disables delivery flag transfer and is the default.

Setting `flagtransfer` is equivalent to setting bit 6 (value 64) of the `deliveryflags` channel option.

33.3.27.9 Limiting time to deliver (`deliverbychannel`)

33.3.27.10 Limiting time to deliver (`deliverbymin`)

The `deliverbymin` channel option controls the availability and the minimum by-time allowed by the DELIVERBY SMTP extension specified in [RFC 2852](#). A value of -1, the default, disables the extension. A value of 0 enables the extension with no minimum by-time. Any other value is treated as the minimum by-time.

33.3.27.11 Solicitation control (`destinationnosolicit`, `sourcenosolicit`)

The NO-SOLICITING SMTP extension described in [RFC 3865](#) provides the means to label messages with solicitation types and for MTAs to block solicitations by type. The `sourcenosolicit` source channel option is used to specify a list of solicitation field values that will be blocked in mail submitted by this `channel`. This list of values will appear in the NO-SOLICITING EHLO response. Glob-style wildcards can be used in the values; however, values containing wildcards will not appear in the EHLO announcement.

The `destinationnosolicit` channel option specifies a list of solicitation field values that will not be accepted in mail queued to this channel.

Note that alternatively, recipient-based no-solicitation settings can be established using the Unified Configuration alias option `alias_nosolicit`, or the `alias file named parameter [NOSOLICIT]`, or the LDAP attributes named by the `ldap_nosolicit` and `ldap_domain_attr_nosolicit` MTA options.

33.3.27.12 SMTP transaction limit (`transactionlimit`, `disconnecttransactionlimit`)

`transactionlimit`: new in Messaging Server 6.1. The `transactionlimit` channel option may be used to impose a limit on how many transactions (that is, messages) will be accepted during a single SMTP session (that is, connection). It is a channel analogue of the `ALLOW_TRANSACTIONS_PER_SESSION` TCP/IP-channel-specific option setting, which applies more generally to all incoming connections on all channels handled by the same Dispatcher [SMTP service](#). After `transactionlimit` is exceeded, additional attempts to submit messages (additional MAIL FROM: commands) will be rejected with an error response:

450 4.5.3 number of transactions exceeds allowed maximum

The text in the above error message may be site-customized by using the (new in JES MS 6.3) `error_text_transaction_limit_exceeded` MTA option.

`disconnecttransactionlimit`: new in JES MS 6.2. The `disconnecttransactionlimit` channel option causes the MTA to actually disconnect after the specified transaction limit is exceeded. Once the limit is reached, the MTA will issue an error response

450 4.7.0 Session transaction limit reached; disconnecting

after the next MAIL FROM or RSET command, and then disconnect.

33.3.27.13 Sending an SMTP VRFY command (`domainvrfy`, `localvrfy`, `novrfy`)

These options control the MTA's use of the VRFY command in its [SMTP client](#). Under normal circumstances there is no reason for the MTA to issue a VRFY command as part of an SMTP dialogue - the SMTP MAIL TO command should perform the same function that VRFY does and return an appropriate error. However, while fairly rare, SMTP servers exist that will accept any address in a MAIL TO (and bounce it later), whereas they perform more extensive checking as part of a VRFY command.

Therefore the MTA can be configured to issue SMTP VRFY commands for each recipient address. The channel option `domainvrfy` causes the MTA to issue a VRFY command with a full address (*e.g.*, `user@host`) as its argument. The `localvrfy` option causes the MTA to issue a VRFY command with just the local part of the address (*e.g.*, `user`). `novrfy` is the default.

Note that while [RFC 1123](#) updated [RFC 821](#) to require support of the VRFY command so modern SMTP servers should have VRFY support, originally [RFC 821](#) did not require that SMTP implementations support VRFY so obsolete SMTP implementations may not have any VRFY support at all. Also note that [RFC 821](#) intentionally left it up to individual implementations to decide on the syntax of the VRFY argument---to decide, that is, what sorts of arguments would get successful responses. So relying on getting a successful response to any sort of VRFY command to determine whether or not to try submitting an address is not, in general, wise. Thus use of `localvrfy` or `domainvrfy` is normally only suitable on special channels sending to known special SMTP hosts with well-understood and special VRFY response behavior.

For controlling the MTA's SMTP server responses to VRFY commands, see the `vrfy*` channel options.

33.3.27.14 Eight bit SMTP capability and EAI capability (`eightbit`, `eightnegotiate`, `eightstrict`, `sevenbit`, `utf8header`, `utf8negotiate`, `utf8strict`)

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. Indeed, there have even been past cases of SMTP servers which will crash when presented with eight bit data.

The MTA 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 on a given channel by specifying the `sevenbit` channel option. A channel should be marked `eightbit` if no such restriction exists.

Some transports such as extended SMTP may actually support a form of negotiation to determine if eight bit characters can be transmitted. The `eightnegotiate` channel option can be used 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` source channel option tells the MTA to reject any messages that contain unnegotiated eight bit data; the exact text of this error may be controlled via the `error_text_unnegotiated_eightbit` MTA option.

33.3.27.15 Responding to SMTP EXPN commands (`expnallow`, `expndefault`, `expndisable`)

New in JES MS 6.1-0.01: These options control, at a channel level, the SMTP server's response when a sending SMTP client issues an SMTP EXPN command. (These channel options do not apply to or affect LMTP servers.) When placed on a source channel, the `expnallow` option tells the SMTP server to issue a detailed, informative response. The `expndefault` tells the SMTP server to provide a detailed, informative response, unless the TCP/IP channel option `DISABLE_EXPAND=1` has been specified. The `expndisable` channel option tells the SMTP server to reject the command with an error:

```
550 5.7.2 EXPN command has been disabled
```

Thus these options allow per-channel control of EXPN responses, as opposed to the `DISABLE_EXPAND` TCP/IP-channel-specific option setting which normally applies to all incoming TCP/IP channels handled via the same SMTP server.

Note that even when EXPN responses are allowed in general, mailing lists or address groups may be configured to disallow EXPN expansion of their membership set of addresses. Such mailing list controls are configured via either the `alias_nonexpandable` alias option (in legacy configuration, the aliases file [NONEXPANDABLE] named parameter), or the LDAP attributes named by the `ldap_expandable` MTA option (normally the `mgmanMemberVisibility` and `expandable` LDAP attributes). The channel restriction on EXPN, if any, is applied first; only if the channel permits EXPN does any mailing list specific restriction get checked.

33.3.27.16 SMTP Future Release Extension (`futurerelease`)

Release 7 of the Messaging Server MTA implements support for future release SMTP SUBMIT extension defined in [RFC 4865](#). This support is enabled by placing the `futurerelease` channel option on the source channel used for initial message submission. The option takes a single integer argument: The maximum number of seconds a message can be deferred.

Care should be used when enabling future release since it allows messages to be in effect stored in the MTA's queues. Future release should only be used for channels handling initial message submission and authentication should be required.

Note that similar functionality is available in earlier Messaging Server releases: Specification of a Deferred-delivery: header field in a submitted message coupled with use of the `deferred` channel option on the destination channel provided the ability to defer delivery of messages. However, future release provides superior functionality:

- The facility is controlled by a setting on the source channel, allowing it to be provided to a subset of the user population. Placing `deferred` on a destination channel opened the door to anyone submitting a message to that channel that would be deferred for some period of time.
- There's no way for a client which sets a Deferred-delivery: header field to know whether or not the header has actually caused the message to be deferred. The future delivery SMTP extension, on the other hand, lets the client know how long a message can be deferred and an error will be returned to the client if the message cannot be deferred for the time the client wants.
- There was no way to place a limit on the amount of time a message could be deferred. Instead what happened was that a message deferred longer than the channel's last `notices` value would simply be returned as undeliverable.
- Deferred-delivery settings on messages did not survive a Job Controller restart.

As part of the implementation work for future release the old Deferred-delivery: mechanism has been redesigned to address some (but not all) of these points. In particular, the `deferred` channel option has been replaced by two new channel keywords, `deferredsource` and `deferreddestination`. (The `deferred` option is now a synonym for `deferreddestination`.) Both of these options accept an integer argument (required in unified configurations, optional in `imta.cnf`) specifying in seconds the maximum amount of time in the future a Deferred-delivery: header can specify and still be honored. The default if no argument is specified is 60*60*24*7, or 7 days. `deferredsource` enables Deferred-delivery: processing on the basis of the source channel while `deferreddestination` operates on destination channels. Finally, Deferred-delivery settings on messages now survive job controller restarts. This addresses all of the points on the above list except the second one - use of a Deferred-delivery: header field still provides no mechanism for informing the client whether or not the setting will be honored.

However, as a purely practical matter, the mechanism chosen to provide delayed release of messages is likely to be dictated by the choice of email client and what mechanisms it supports.

33.3.27.17 Channel protocol selection (`smtp`, `smtp_cr`, `smtp_crlf`, `smtp_crorlf`, `smtp_lf`, `nosmtp`, `lmtp`, `lmtp_cr`, `lmtp_crlf`, `lmtp_crorlf`, `lmtp_lf`)

These channel options specify whether or not a channel supports the SMTP protocol (or LMTP protocol) and what type of SMTP line terminator (or LMTP line terminator) the MTA expects to see as part of that protocol. `nosmtp` means that the channel doesn't support either SMTP or LMTP; all the rest of these channel options imply either SMTP or LMTP support.

The selection of whether or not to use the SMTP or LMTP protocol is implicit for most channels; the correct protocol is chosen by the use of the appropriate channel program or programs.

The channel option `smtp`---or one of the `smtp_*` variants--- is mandatory for all SMTP channels. The channel option `lmtp`---or one of the `lmtp_*` variants---is mandatory for all LMTP channels.

The channel options `smtp_cr`, `smtp_crlf`, `smtp_crorlf`, and `smtp_lf` may be used on SMTP channels to specify what character sequences to accept as line terminators. `smtp` or `smtp_crlf` means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. `smtp_crorlf` means that lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF. (Note that prior to JES MS 6.0, `smtp` used to be synonymous with `smtp_crorlf`, rather than with `smtp_crlf` as currently; this change to a more strict insistence on proper SMTP line terminators was made in accordance with [RFC 2821](#).) `smtp_lf` means that a LF without a preceding CR will be accepted. Finally, `smtp_cr` means that a CR will be accepted without a following LF. It is normal to use CRLF sequences as the SMTP line terminator, and this is what the MTA itself always generates; this option only affects the MTA's handling of incoming material.

The `lmtp*` channel options are similar, applying to the LMTP protocol rather than the SMTP protocol.

Note that the setting of the original, "default" incoming TCP/IP channel is what controls the behavior for all incoming TCP/IP channels to which that channel may subsequently "switch". That is, subsequent "switching" (due, for instance, to [switchchannel](#), [saslsplitchannel](#), [tlssplitchannel](#), or `mailSMTPSubmitChannel` sorts of effects) will not result in a change of SMTP line terminator regardless of what may be set on that "switched to" channel; the option specified on the original incoming TCP/IP channel (typically `tcp_local`) stays in effect.

33.3.27.18 The XLOOP SMTP extension for blocking message loops (loopcheck, noloopcheck)

The SMTP server unconditionally includes the XLOOP extension and an identifying string in its EHLO response.

Specifying the `loopcheck` channel option tells the SMTP client to make use of XLOOP if advertised by a "remote" server. An SMTP client can then check a hash (of the configuration file), to compare with that advertised by the SMTP server to which it is connected to see if it, too, is on the same system. If so, the SMTP client bounces the message, generating a notification message with a "SMTP client-server loop detected" reason and a "5.4.6 (SMTP client-server loop detected)" error status in the notification message. Note that this rejection is done by the SMTP client itself, immediately upon processing the SMTP server's EHLO response. Thus using `loopcheck` causes certain sorts of looping messages to be immediately bounced, rather than looping until they become `.HELD` files.

In terms of logging of such cases if the `logging` channel option is used, note that the SMTP server does not generate a "J" record rejecting the message, since the SMTP server did not in fact reject the message, but rather it was the SMTP client that decided to abort sending of the message. And the SMTP client's "R" record for the rejection of the message(s) does not show an SMTP error (such as the 5.4.6 error shown in the notification message itself) issued from the SMTP server; (the SMTP server did not in fact issue that error). The fact that it was a rejection due to `loopcheck` is instead implicit in the fact that the host connected to was the same as the SMTP client host. This may be seen via the transport information in the "R" record, if bit 1 (value 2) of the `log_connection` MTA option was enabled.

`noloopcheck` is the default.

See also the (new in 6.3) [IP_ACCESS mapping table](#), which provides an alternate way to block connecting to specified destination IP addresses (*e.g.*, 127.0.0.1).

33.3.27.19 Verify that the domain on the MAIL FROM: line is in the DNS (`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. `nomailfromdnsverify` is the default, and 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 junk e-mail (SPAM) is being sent with forged e-mail addresses from non-existent domains.

The introduction of DNS wildcard entries in the COM and ORG top level domains which occurred in September 2003 severely limited the effectiveness of the `mailfromdnsverify` channel option. (The wildcards have subsequently been removed, however, such practices could resume at any time.) As of the 6.1 release of the Messaging Server MTA, `mailfromdnsverify` code has been modified to address this. When the DNS returns one or more A records (which would normally be considered a "success" and the message would be allowed in), their values are compared against the domain literals specified by the MTA option [blocked_mail_from_ips](#). If a match is found, then the domain is considered to be invalid.

With `mailfromdnsverify` on, as of Messaging Server 6.0 and later the MTA attempts an MX lookup on the domain of the MAIL FROM: command. As of JES MS 6.1 and later, if that MX lookup returns no data (no MX record exists) then the MTA moves on to attempting a `gethostbyname` call. That is, a success at the MX record lookup stage allows the message in; errors other than simply no such MX record (*e.g.*, a nameserver "server failed" error) at this MX record lookup stage will result in a temporary rejection with error

```
450 4.1.8 invalid/host-not-in-DNS return address not allowed
```

while (with JES MS 6.1 or later) a no such MX record found case moves onward to checking the result of a `gethostbyname` call. (In iMS 5.2, only the `gethostbyname` was attempted; no explicit MX record lookup was performed.)

When the MTA does a `gethostbyname` call, if this DNS query results in an authoritative "host not found" response, then the message will be rejected with a permanent rejection

```
550 5.1.8 invalid/host-not-in-DNS return address not allowed
```

error message. A no data response, as would occur for the case of a name which has only a CNAME record in the DNS, is considered a successful response; the message will be allowed in. Any other error responses from the DNS will result in a temporary error

450 4.1.8 invalid/host-not-in-DNS return address not allowed

deferring the message: the MTA will not accept the message at the present time, but the sending side should try sending it again later (in case perhaps their DNS problem, whatever it was, gets fixed).

New in 8.0 is specialized handling for MX entries of the form:

```
nomail          IN MX 0      .
```

Such entries are intended to be an indication that host "nomail" does not operate a mail server. Support has been added so that `mailfromdnsverify` will treat such hosts as not being a valid source of mail. (Additionally, attempts to send to such a host will fail immediately after the [MX lookup](#) instead of attempting any sort of A record lookup.)

If the [logging](#) channel option has been enabled on an incoming channel, then rejections due to a `mailfromdnsverify` check on that channel will be logged to the `mail.log*` file as a "J" record.

33.3.27.20 Microsoft Exchange gateway channels (`msexchange`, `nomsexchange`)

The `msexchange` channel option may be used on [TCP/IP channels](#) to tell the MTA that this is a channel that communicates with Microsoft[®] Exchange gateways and clients. Use of the option tells the MTA to try and accommodate nonstandard behavior on the part of Microsoft Exchange. Exactly what nonstandard behaviors are dealt with is subject to change.

Currently the `msexchange` channel option on a channel configured to allow TLS use (see the [*tls* channel options](#)) causes advertisement (by the MTA's SMTP server) and recognition (by the MTA's SMTP client) of the non-standard TLS capability string, in addition to the standard STARTTLS capability string, to indicate that TLS is supported.

New in 7.0.5, setting `msexchange` on a destination channel will cause the MTA, if performing any sort of MIME processing operation, to remove any Content-disposition: header line from any text/calendar message parts, as despite Content-disposition:'s long-standing existence as a standardized header line, not to mention the basic MIME rule that unrecognized Content-* header lines should be ignored, Microsoft[®] Outlook's handling of text/calendar parts is disturbed when such parts have a Content-disposition: specified. So specifying `msexchange` on a channel sending to Microsoft Exchange, if text/calendar parts will flow through that channel, should allow Microsoft Outlook to process calendar parts more successfully.

`nomsexchange` is the default.

33.3.27.21 Per-channel MT-PRIORITY control (`mtprioritiesallowed`, `mtprioritiesrequired`)

`mtprioritiesallowed` and `mtprioritiesrequired` are new in the 8.0 release. These channel options enable the MTA's support of [RFC 6710 \(SMTP Extension for Message Transfer Priorities\)](#).

The `mtprioritiesallowed` source channel option specifies the range of MT-PRIORITY values that will be accepted. MT-PRIORITY values outside this range will be adjusted up or down so they fall within the allowed range. If a single argument is given, it specifies the

highest priority value that will be accepted. The default if this option is not specified is for the MT-PRIORITY extension not to be offered and for MT-PRIORITY parameters not to be accepted.

The `mtprioritiesrequired` source channel option specifies the range of MT-PRIORITY that will be accepted for enqueue. If a single argument is given, it specifies the lowest priority value that will be accepted. The message will be rejected if the message's specified MT-PRIORITY value, or if the default MT-PRIORITY value of 0 (assumed if MT-PRIORITY was not specified in the SMTP transaction), falls outside the required range with the SMTP error:

```
550 5.7.0 Message priority outside curretly allowed range
```

With either channel option, two integer arguments specify the range. Each argument must be an integer in the range -9..9. The arguments can be given in any order.

33.3.27.22 SMTP DSN extension support (`notary`, `refusenotary`, `nonotary`)

The `notary` and (the RESTRICTED) `nonotary` channel options control whether client [TCP/IP channels](#) attempt to use the SMTP DSN extension (defined in [RFC 3461](#)). The `notary` channel option is the default on SMTP over TCP/IP channels.

The `nonotary` channel option, if set, disables the use of the SMTP NOTARY extension. Its use on SMTP client channels is RESTRICTED, and it is normally used only on LMTP client channels. Note that setting `lmtp` or an `lmtp_*` channel option on a channel implicitly sets `nonotary`.

New in 8.0.1 is the `refusenotary` channel option. This RESTRICTED option disables the DSN extension in the SMTP/LMTP client and additionally, the SMTP server. (The DSN extension is never offered by the LMTP server.)

33.3.27.23 Sending an SMTP ETRN command (`sendetrn`, `nosendetrn`)

The extended SMTP command ETRN ([RFC 1985](#)) allows an SMTP client to request that a remote SMTP server start up processing of the remote side's message queues destined for sending to the original SMTP client; that is, it allows an SMTP client and SMTP server to negotiate "switching roles", where the side originally the sender becomes the receiver, and the side originally the receiver becomes the sender. Or in other words, 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 only have transient connections between each other, for instance, over dial up lines. When the connection is brought up and one side sends to the other, via the ETRN command the SMTP client can also tell the remote side that it should now try to deliver any messages that need to travel in the reverse direction.

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.

The `sendetrn` and `nosendetrn` channel options control whether the [SMTP client](#) 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` channel option tells the MTA to send an ETRN command, if the remote SMTP server says it supports ETRN. The `sendetrn` requires an argument giving the name of the system requesting that its messages receive a delivery attempt to send in the ETRN command.

33.3.27.24 SMTP TURN command channel options (`noturn`, `turn`, `turn_in`, `turn_out`)

[RFC 821](#) defined an optional TURN command, for the client (sender) and server (receiver) to switch roles. It is not normally appropriate to enable use of TURN: it is quite dangerous, as it allows any arbitrary client to "snatch" your messages! Use of the default `noturn` is thus STRONGLY RECOMMENDED!

For a safer "relay upon demand" feature, see the ATRN SMTP extension ([RFC 2645](#)).

33.3.27.25 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA provides support for Postfix's XCLIENT SMTP extension. The PostFix documentation for the extension can be found here:

http://www.postfix.org/XCLIENT_README.html

Use of XCLIENT is controlled by three main source channel keywords, `noxclient`, `xclient`, and `xclientsasl`, and variants `xclientrepeat` and `xclientsaslrepeat`. `noxclient` is the default, and means that XCLIENT is not advertised in the response to EHLO and the XCLIENT command itself is disabled. If `xclient` is set the XCLIENT command is enabled and the NAME, ADDR, PORT, PROTO, and HELO attributes may be used. `xclientsasl` enables the LOGIN attribute in addition to all the others. It should be noted that LOGIN specifies an external identity that must then be bound to the session identity through the use of SASL EXTERNAL.

By default, only one set of XCLIENT commands is allowed in a single SMTP session. Specifying `xclientrepeat` allows groups of XCLIENT commands to be repeated, allowing a proxy or similar agent to share a connection between multiple clients. `xclientsaslrepeat` allows multiple groups of XCLIENT commands including LOGIN. Note that care should be taken when these keywords are used since the server cannot determine the origin of a given XCLIENT command.

The primary visible effect of XCLIENT is on the contents of the Received: field the MTA adds. For example, if this XCLIENT command was executed:

```
xclient name=foo.domain.com addr=1.2.3.4 helo=bar.domain.com port=12345
```

it would result in a header of the general form:

```
Received: from bar.domain.com (foo.domain.com [1.2.3.4])
  by server.domain.com (Oracle Communications Messaging Server 7.0.5.32
  64bit (built Aug 18 2014)) with imapsubmit
  id <010J9P51WPFC007KNZ@server.domain.com> for user@domain.com;
```

Mon, 20 Aug 2012 08:17:31 -0700 (PDT)

However, the ADDR and PORT attributes also change the contents of the transportinfo that appears in various mapping table probes, such as the probe to [PORT_ACCESS](#). Given the preceding XCLIENT command, the transportinfo part of the mapping probes would change to something like:

```
TCP | <destaddr> | 25 | 1.2.3.4 | 12345
```

33.3.27.26 SMTP long line handling (`rejectsmtplonglines`, `truncatesmtplonglines`, `wrapsmtplonglines`)

The SMTP protocol is a line oriented protocol, and in particular, SMTP transmissions are limited to a maximum of 1000 characters including the carriage-return CR and line-feed LF characters, or 998 characters not including the CRLF sequence; (though see [RFC 3030](#) for a proposed extension of the SMTP protocol that relaxes this definition). Nevertheless, there are some clients (typically arising in HTML applications) that try to send illegally long lines over SMTP.

The channel options `truncatesmtplonglines`, `rejectsmtplonglines`, and `wrapsmtplonglines` control the MTA's behavior when it sees such illegal long lines in incoming SMTP messages. `truncatesmtplonglines` is the default, and causes the MTA to truncate illegally long SMTP lines to the legal length limit; the MTA also in JES MS 6.0 inserts a header line

```
Sun-ONE-SMTP-Warning: Lines longer than SMTP allows found and truncated.  
or in JES MS 6.1 or later, a header line
```

```
Sun-Java-System-SMTP-Warning: Lines longer than SMTP allows found and truncated.
```

when it sees such long lines. The `rejectsmtplonglines` option causes the MTA to reject such illegal messages with a

```
550 5.6.0 lines longer than SMTP allows encountered; message rejected
```

SMTP error, and may be useful when it is desired to enforce strict standards compliance upon message submissions. The `wrapsmtplonglines` keyword causes the MTA to forcibly wrap (insert hard line breaks) into illegally long incoming lines, and insert a header line of (in JES MS 6.0)

```
Sun-One-SMTP-Warning: Lines longer than SMTP allows found and wrapped.
```

```
or in JES MS 6.1 or later
```

```
Sun-Java-System-SMTP-Warning: Lines longer than SMTP allows found and wrapped.
```

The MTA will attempt to find a space or TAB character in the line as a suitable place at which to perform the line break. In the absence of such a white space character, the MTA may have

to add the hard line break at a less suitable location, changing the character of the data; in particular, `wrapsmtplonglines` when applying to a header line (to an illegally long line in a message header) may damage the message header since the forced line break may (in the absence of white space characters) cause a syntactically illegal line. `wrapsmtplonglines` is hence only intended for dealing with cases of illegally long lines in the message body. Furthermore, some charsets (e.g., ISO-2022-JP) have encoding requirements that are triggered at line wraps, so that forcible line wrapping can interact badly with such charsets; or even when a message body is in a charset that has no line wrap/encoding issues, the message content itself may be "damaged" by line wrapping. `wrapsmtplonglines` is thus only a partial workaround to cases of illegally long data in SMTP transmitted messages; it makes an attempt to more-or-less preserve message contents, but some damage is not unexpected.

Note that these channel options must be placed on the initial (default) incoming [TCP/IP channel](#) in order to take effect, that is, the SMTP server default channel; for instance, these options would typically be used on a `tcp_local` or `tcp_submit` channel. These channel options do not take effect on a channel "switched to" subsequently (due for instance to a [switchchannel](#), [tlsswitchchannel](#), or [saslswitchchannel](#) channel option effect, or a `mailSMTPSubmitChannel` LDAP attribute effect).

33.3.27.27 Protocol streaming (`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` channel option controls the degree of protocol streaming used in the protocol associated with a channel. This option requires an integer argument; how the argument is interpreted is specific to the protocol in use.

Currently the MTA only supports the use of streaming on SMTP channels. Streaming is enabled automatically for the MTA's SMTP client if the SMTP server to which the MTA has connected offers the pipelining extension and the streaming setting is nonnegative. The `streaming` option can be used to enable (force) streaming by the MTA's SMTP client even when a remote SMTP server doesn't offer the pipelining extension.

The streaming values available for SMTP range from -2 to 4. Negative values (new in 7.2-7.02) disables streaming completely; not even the remote SMTP server advertising pipelining can enable it. 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 to stream, a value of 3 causes HELO/MAIL FROM/RCPT TO or RSET/MAIL FROM/RCPT TO streaming to be used, and a value of 4 enables streaming all the way through DATA (equivalent to the remote server advertising pipelining). The default value is 0.

The SMTP server offers the pipelining extension by default. A streaming value of -2 (new in 7.2-7.02) can be used to disable pipelining announcement.

Some SMTP implementations are known to react badly to streaming. In particular, many versions of sendmail are known to be incapable of handling streaming levels greater than 1. The MTA's server implementation of SMTP should work properly at any streaming level.

New in Messaging Server 7.0, [MTA message transaction log entries](#) will record whether PIPELINING was used by means of a "Q" modifier on the relevant "E" (Enqueue) and "D" (Dequeue) entries.

33.3.27.28 Responding to SMTP VRFY commands (`vrfyallow`, `vrfydefault`, `vrfyhide`)

These channel options control the MTA's SMTP server's response when a sending SMTP client issues an SMTP VRFY command. The `vrfyallow` channel option tells the SMTP server to issue a detailed, informative response. The `vrfydefault` option tells the SMTP server to provide a detailed, informative response, unless the TCP/IP-channel-specific option `HIDE_VERIFY=1` has been specified. The `vrfyhide` option tells the MTA to issue only a vague, ambiguous response. Thus these channel options allow per-channel control of VRFY responses, as opposed to the `HIDE_VERIFY` TCP/IP-channel-specific option which normally applies to all incoming TCP/IP channels handled via the same SMTP server.

For controlling the MTA's SMTP client use of VRFY commands, see the `*vrfy` channel options.

33.3.28 TCP/IP connections and DNS lookups channel options

A number of channel options affect TCP/IP connections and DNS lookups.

33.3.28.1 Channel connection information caching (`cacheeverything`, `cachesuccesses`, `cachefailures`, `nocache`)

Channels using the [SMTP and LMTP protocols](#) maintain a per-process 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 only lasts for the duration of the delivery process; subsequent processes start with an empty cache.

(In this context, "connection failure" includes both [connection transaction log file](#) "Y" entries -- where the MTA's client couldn't even make a connection -- and at least some "X" entries, such as connecting but immediately seeing a 5xx or 4xx error instead of an SMTP banner.)

Such a process 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. For example, a channel that is used to connect to a single flakey host does not benefit from caching. Or in the case of connection attempts to an internal LMTP "back end", it may be desirable to continue to try to connect regardless of previous connection failures. Therefore channel options are provided to adjust the MTA's SMTP and LMTP client process caches.

The `cacheeverything` channel option enables all forms of caching and is the default. `nocache` disables all caching. `cachefailures` enables caching of connection failures but not successes --- this forces a somewhat more draconian retry than `cacheeverything` does. Finally, `cachesuccesses` caches only successes. This last option is effectively equivalent to `nocache` for SMTP and LMTP channels.

In the [MTA message transaction log file](#), "Q" entries with the notation:

```
Too many failures to this host during this run; skipping this host: error-text
```

are cases where a process cache had come into play.

33.3.28.2 Envelope address rewriting upon message dequeue (connectalias, connectcanonical)

The MTA normally rewrites addresses as it enqueues messages to its channel queues. No additional rewriting is done 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` option tells the MTA to simply deliver to whatever host is listed in the recipient address. This is the default. `connectcanonical` tells the MTA to compare the recipient envelope address domain with the channel host proper names, and if the domain name matches one of the channel's host proper names, then connect to the host name corresponding to that host proper name. For instance, if a channel is defined (in `pmdf.cnf/imta.cnf` format) as:

```
tcp_scanner smtp connectcanonical ...rest-of-keywords...  
SCANNER-DAEMON  
scanner1.domain.com host1.domain.com  
scanner2.domain.com host2.domain.com
```

then an address of `user@host1.domain.com` that rewrites to the `tcp_scanner` channel will be routed out to the host `scanner1.domain.com`, rather than to `host1.domain.com` as would occur by default.

`connectcanonical` should only be used specifically to deal with problems with queued messages--it may have unintended effects on other message traffic.

33.3.28.3 Forced routing to gateways (daemon)

The interpretation and usage of the `daemon` channel option depends upon the type of channel to which it is applied. Currently, the only type of channel for which the `daemon` option is relevant is SMTP over [TCP/IP channels](#). Normally such channels connect to whatever host is listed in the envelope address of the message being processed. The `daemon` option is used to tell the channel to instead connect to a specific remote system, generally a firewall or mailhub system, regardless of the envelope address. The actual remote system name is given as an argument to `daemon`, *e.g.*:

```
msconfig> set channel:tcp_firewall.daemon firewall.domain.com
```

If the argument after the `daemon` option is not a fully qualified domain name (or alternatively a square bracket enclosed literal IP address), the argument will be ignored and the channel will connect to the channel's official host. When specifying the firewall or gateway system name as the channel's [official host name](#), `channel:channel-name.official_host_name`, the argument given to the `daemon` option is typically specified as `router`, *e.g.*:

```
msconfig> show channel:tcp_firewall  
role.channel:tcp_firewall.official_host_name = firewall.domain.com  
role.channel:tcp_firewall.daemon = router  
role.channel:tcp_firewall.mx (novalue)  
role.channel:tcp_firewall.pool = SMTP_POOL  
role.channel:tcp_firewall.smtp (novalue)
```

33.3.28.4 TCP/IP nameserver and MX record support (`mx`, `nomx`, `nodns`, `defaultmx`, `randommx`, `nonrandommx`, `affinitylist`, `nameservers`, `defaultnameservers`)

Most TCP/IP networks support the use of MX (mail forwarding) records for SMTP relay but a few do not. The MTA's [TCP/IP channel](#) programs can be configured to not use MX records if they are not provided by the network to which the MTA system is connected. Some TCP/IP channel programs can be configured to not do DNS (nameserver) lookups at all. `randommx` specifies that MX lookups should be done and MX record values of equal precedence should be processed in random order. `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` option is currently equivalent to `nonrandommx`; it may change to be equivalent to `randommx` in a future release. The `nomx` option disables MX lookups. The `defaultmx` option specifies that `mx` should be used (with randomization) if the network says that that MX records are supported, and if the destination port is port 25 for SMTP; (with `defaultmx`, destination ports other than port 25 do not get MX lookups by default).

LMTP channels do not normally make use of MX lookups. Additionally, LMTP channels need to connect to the correct store taking failover events into account. As of the 8.0 release, the `affinitylist` channel option provides this functionality. `affinitylist` disables MX lookups completely and translates the logical host given into the corresponding affinity group. Connections are then attempted sequentially to all the hosts in the group.

The default is `defaultmx` on channels that support MX lookups in any form.

New in 8.0 is specialized handling for MX entries of the form:

```
nomail          IN MX 0      .
```

Such entries are intended to be an indication that host "nomail" does not operate a mail server. So when MX lookups are enabled, attempts to send to such a host will fail immediately after the MX lookup instead of attempting any sort of A record lookup. (Additionally, [mailfromdnsverify](#) will treat such hosts as not being a valid source of mail.)

On UNIX, whether the underlying TCP/IP package's local host tables are used in addition to the DNS for lookups is up to the underlying TCP/IP package configuration. Generally, TCP/IP packages are configured so that local host tables will indeed be consulted. Consult your TCP/IP package documentation for details. (On Unix, the `nomx` and `nodns` channel options are effectively the same; on Unix, `nodns` merely disables MX lookups, and does not control whether nameserver lookups are performed.)

When nameserver lookups are being performed, that is, unless the `nodns` channel option is used on OpenVMS, or the `nsswitch.conf` file on UNIX selects no use of nameservers, then prior to Messaging Server 7.0 the `nameservers` channel option may be used to specify a list of nameservers to consult rather than consulting the TCP/IP stack's own choice of nameservers. This affects the SMTP server and client and LMTP client, but *not* the LMTP server (which, if it needs to do any lookups, always relies on the TCP/IP stack's own choice of nameservers). Furthermore, as of Messaging Server 7.0, the `nameservers` channel option only affects MX record lookups, with all other lookups using the TCP/IP stack's choice of nameservers regardless of any `nameservers` channel option setting. `nameservers` requires a space separated list of IP addresses for the nameservers, *e.g.*,

1.2.3.1 1.2.3.2

`defaultnameservers` is the default, and means to use the TCP/IP stack's own choice of nameservers.

Note that while a `nameservers` setting is primarily meaningful to the SMTP client -- hence TCP/IP destination channels -- it also, prior to Messaging Server 7.0, potentially had meaning to the SMTP server -- hence TCP/IP source channels -- as for instance in cases where the SMTP server channel had been configured to perform DNS reverse lookups on incoming connections, or to perform forms of DNS verification.

33.3.28.5 Reverse DNS and IDENT lookups on incoming SMTP connections (`identtcp`, `identtcplimited`, `identtcpnumeric`, `identtcpsymbolic`, `identnone`, `identnonelimited`, `identnonenumeric`, `identnonenumeric`, `forwardchecknone`, `forwardchecktag`, `forwardcheckdelete`)

The `identtcp` channel option tells the MTA to perform a connection and lookup using the IDENT protocol (RFC 1413). The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is then inserted into the Received: header for the message, with the hostname corresponding to the incoming IP number, as reported from a DNS reverse lookup, and the IP number itself.

The `identtcpsymbolic` channel option tells the MTA to perform a connection and lookup using the IDENT protocol (RFC 1413). The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is then inserted into the Received: header for the message, with the hostname corresponding to the incoming IP number, as reported from a DNS reverse lookup; the IP number itself is not included in the Received: header.

The `identtcpnumeric` channel option tells the MTA to perform a connection and lookup using the IDENT protocol (RFC 1413). The information obtained from the IDENT protocol (usually the identity of the user making the SMTP connection) is then inserted into the Received: headers of the message, with the actual incoming IP number --- no DNS reverse lookup on the IP number is performed.

Note that the remote system must be running an IDENT server in order for the IDENT lookup caused by the `identtcp`, `identtcpsymbolic`, or `identtcpnumeric` options to be useful. In addition, be aware that IDENT query attempts may incur a serious performance hit. Increasingly routers simply "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 package controlled timeout, typically on the order of a minute or two). A lesser performance factor is that when comparing `identtcp` or `identtcpsymbolic` vs. `identtcpnumeric`, note that the DNS reverse lookup called for with `identtcp` or `identtcpsymbolic` incurs some additional overhead to obtain the more "user-friendly" hostname.

The `identnone` channel option disables this IDENT lookup, but does do IP to hostname translation, and both IP number and hostname will be included in the Received: header for the message. The `identnonenumeric` channel option disables this IDENT lookup, but does do IP to hostname translation; only the hostname will be included in the Received: header for the message. The `identnonenumeric` channel option disables this IDENT lookup and inhibits the usual DNS reverse lookup translation of IP number to hostname, and may therefore result

in a performance improvement at the cost of less user-friendly information in the Received: headers. `identnone` is the default.

The `identtcplimited` and `identnonelimited` channel options have the same effect as `identtcp` and `identnone`, respectively, as far as IDENT lookups, reverse DNS lookups, and information displayed in Received: header lines. Where they differ is that with `identtcplimited` or `identnonelimited` the IP literal address is always used as the sole basis for any channel switching due to use of the `switchchannel` channel option, regardless of whether the DNS reverse lookup succeeds in determining a host name. Note that since channel switching is always performed preferentially based on IP address rather than host name, the effect of `identtcplimited` or `identnonelimited` is merely to disable ever trying host name switching in case all IP address rewriting failed.

Table 33.15 Available `ident*` MTA options and interpretations

Channel option	IDENT lookup	DNS reverse lookup	IP address in Received: header line	Reverse hostname in Received: header line	Fall back to hostname channel switch
<code>identtcp</code>	Yes	Yes	Yes	Yes	Yes
<code>identtcplimited</code>	Yes	Yes	Yes	Yes	No
<code>identtcpnumeric</code>	Yes	No	Yes	No	No
<code>identtcpsymbolic</code>	Yes	Yes	No	Yes	Yes
<code>identnone</code>	No	Yes	Yes	Yes	Yes
<code>identnonelimited</code>	No	Yes	Yes	Yes	No
<code>identnonenumeric</code>	No	No	Yes	No	No
<code>identnonesimalic</code>	No	Yes	No	Yes	Yes

The `forwardchecknone`, `forwardchecktag`, and `forwardcheckdelete` channel options can modify the effects of doing reverse lookups, controlling whether the MTA does a forward lookup of an IP name found via a DNS reverse lookup, and if such forward lookups are requested, further control what the MTA does in case the forward lookup of the IP name does not match the original IP number of the connection. The `forwardchecknone` channel option is the default, and means that no forward lookup is done. The `forwardchecktag` channel option 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 via the forward lookup does not match that of the original connection. The `forwardcheckdelete` channel option 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, and stick with the original IP address instead. (Note that 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.)

These options are only useful on [SMTP channels that run over TCP/IP](#).

33.3.28.6 TCP/IP interface address (`interfaceaddress`)

The `interfaceaddress` channel option 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 channel option controls which address will be used as the source

IP address when the MTA sends outgoing SMTP messages. Note that it complements the Dispatcher option `listenaddr` (INTERFACE_ADDRESS in legacy configuration), which controls which interface address a TCP/IP channel's SMTP server program listens on for accepting incoming connections and messages. Also note that such channel `interfaceaddress` settings are quite separate from the Job Controller's own `listenaddr` setting (INTERFACE_ADDRESS setting in legacy configuration), which merely controls what IP address the Job Controller listens on for purposes of its own, internal communications.

33.3.28.7 Specify a last resort host for delivery (`lastresort`)

The `lastresort` channel option is used to specify a host to which to connect when all other connection attempts fail. In effect this acts as an MX record of last resort. This is only useful on [SMTP over TCP/IP channels](#).

Note that the `lastresort` host is attempted only for hosts that are in the DNS, having either MX records or an A record, and for whom the connection attempts to all the MX records -- or to the A record, if there were no MX records--have encountered temporary connection failures. (In particular, the `lastresort` host will not be attempted for a host that is only in the hosts file, not in the DNS at all. Also keep in mind that a permanent 5xx error in response to a connection attempt to a host is a permanent error, and will result in bouncing a message; in particular, the `lastresort` host will not be attempted after such a permanent rejection error.

Also, the `lastresort` host will not be attempted if a connection succeeds, but the MTA's wait for an SMTP banner line to be returned times out; that again is not a temporary *connection* failure.)

This channel option requires a single parameter specifying the name of the "system of last resort".

See also the [IP_ACCESS mapping table](#), which can provide an alternate way of doing "fail over" for outbound IP connections for SMTP and LMTP channels.

Note that in most cases, it is preferable to fix problematic DNS records rather than to use `lastresort`; `lastresort` is intended merely for a few, special sorts of cases where correcting DNS records may not be possible, yet some "last ditch", MX-like, re-routing may be useful.

33.3.28.8 Verify that the domain on the MAIL FROM: line is in the DNS (`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. `nomailfromdnsverify` is the default, and 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 junk e-mail (SPAM) is being sent with forged e-mail addresses from non-existent domains.

The introduction of DNS wildcard entries in the COM and ORG top level domains which occurred in September 2003 severely limited the effectiveness of the `mailfromdnsverify`

channel option. (The wildcards have subsequently been removed, however, such practices could resume at any time.) As of the 6.1 release of the Messaging Server MTA, `mailfromdnsverify` code has been modified to address this. When the DNS returns one or more A records (which would normally be considered a "success" and the message would be allowed in), their values are compared against the domain literals specified by the MTA option `blocked_mail_from_ips`. If a match is found, then the domain is considered to be invalid.

With `mailfromdnsverify` on, as of Messaging Server 6.0 and later the MTA attempts an MX lookup on the domain of the MAIL FROM: command. As of JES MS 6.1 and later, if that MX lookup returns no data (no MX record exists) then the MTA moves on to attempting a `gethostbyname` call. That is, a success at the MX record lookup stage allows the message in; errors other than simply no such MX record (*e.g.*, a nameserver "server failed" error) at this MX record lookup stage will result in a temporary rejection with error

```
450 4.1.8 invalid/host-not-in-DNS return address not allowed
```

while (with JES MS 6.1 or later) a no such MX record found case moves onward to checking the result of a `gethostbyname` call. (In iMS 5.2, only the `gethostbyname` was attempted; no explicit MX record lookup was performed.)

When the MTA does a `gethostbyname` call, if this DNS query results in an authoritative "host not found" response, then the message will be rejected with a permanent rejection

```
550 5.1.8 invalid/host-not-in-DNS return address not allowed
```

error message. A no data response, as would occur for the case of a name which has only a CNAME record in the DNS, is considered a successful response; the message will be allowed in. Any other error responses from the DNS will result in a temporary error

```
450 4.1.8 invalid/host-not-in-DNS return address not allowed
```

deferring the message: the MTA will not accept the message at the present time, but the sending side should try sending it again later (in case perhaps their DNS problem, whatever it was, gets fixed).

New in 8.0 is specialized handling for MX entries of the form:

```
nomail          IN MX 0      .
```

Such entries are intended to be an indication that host "nomail" does not operate a mail server. Support has been added so that `mailfromdnsverify` will treat such hosts as not being a valid source of mail. (Additionally, attempts to send to such a host will fail immediately after the MX lookup instead of attempting any sort of A record lookup.)

If the `logging` channel option has been enabled on an incoming channel, then rejections due to a `mailfromdnsverify` check on that channel will be logged to the `mail.log*` file as a "J" record.

33.3.28.9 SOCKS connections (`nosocks`, `socksnoauth`, `socksuserpassword`)

SOCKS connections (see [RFC 1928](#) and [RFC 1929](#)) can be used to traverse a firewall that would not normally permit outbound SMTP message traffic. If the firewall offers a SOCKS service, then one can connect to the firewall's SOCKS server and authenticate, pass over the remote host name and remote port to which one wishes to make an SMTP connection, and then the SOCKS server on the firewall will make the remote connection and transform the SOCKS connection into the desired SMTP connection. The `nosocks` and `socksuserpassword` channel options control whether a TCP/IP channel uses a SOCKS connection, rather than attempting a normal, direct SMTP connection. `nosocks`, the default, specifies that no SOCKS connection will be used. `socksuserpassword` tells the channel to attempt a SOCKS connection (using the username/password method of SOCKS authentication), rather than attempting a direct SMTP connection.

To achieve a SOCKS connection, one must set additional options. In legacy configuration, one must set TCP/IP-channel-specific options specifying the SOCKS host and port to which to connect, and the username and password with which to authenticate the SOCKS connection; see the `SOCKS_HOST`, `SOCKS_PORT`, `SOCKS_USERNAME`, and `SOCKS_PASSWORD` channel settings. In Unified Configuration, one must instead set channel options `sockshost`, `socksport`, `socksusername`, and `sockspassword`.

Note that the SOCKS protocol is a general protocol for TCP/IP-based applications (*e.g.*, SMTP), and makes no provision for application-specific issues such as, in the case of SMTP, MX host name DNS lookups. Thus when using a `socksuserpassword` channel, one must be sure that the host names that the channel is attempting to send messages to are all fully resolved hostnames (A record names). As such, `socksuserpassword` channels are all, in effect, `nomx` channels, and hence should only be used for special, point-to-point connections to known, specific remote systems where the proper mail host name is "known" (need not be looked up as a possible MX record in the DNS). In particular, any rewrite rules that direct domains to a `socksuserpassword` channel should output only domain names that are proper mail destination host systems (fully resolved domain names, with any MX references already taken into account).

33.3.28.10 `port` Option Under channel

[SMTP over TCP/IP channels](#) normally connect to the remote system's port 25 when sending messages. The `port` channel option may be used to instruct an SMTP over TCP/IP channel to connect to a non-standard port.

33.3.28.11 SOCKS connections channel options: `sockshost` (host), `socksport` (port), `socksusername` (string) `sockspassword` (string)

The `sockshost`, `socksport`, `socksusername`, `sockspassword` channel options (for SMTP client channels) are used to configure SOCKS connections. SOCKS connections (see [RFC 1928](#) and [RFC 1929](#)) can be to traverse a firewall that would not normally permit outbound SMTP message traffic. If the firewall offers a SOCKS service, then one can connect to the firewall's SOCKS server (`sockshost` and `socksport`) and authenticate (`socksusername` and `sockspassword`), pass over the remote host name and remote port to which one wishes to make an SMTP connection, and then the SOCKS server on the firewall will make the remote connection and transform the SOCKS connection into the desired SMTP connection. The `sockshost` option specifies the host name of the SOCKS server system. The `socksport` option specifies the SOCKS port on the SOCKS server system; by convention, port 1080 is usually used as the SOCKS port. The MTA's SOCKS implementation currently only supports the username/password method of SOCKS authentication; the username and password

to be used are controlled by the `socksusername` and `sockspassword` channel options, respectively.

The `sockshost`, `socksusername`, and `sockspassword` options have no default; the default for the `socksport` option is 1080.

In order for these `socks*` channel options to take effect, the outbound TCP/IP channel must also be marked with the `socksuserpassword` channel option.

33.3.28.12 SPF DNS lookups (`spfhello`, `spfmailfrom`, `spfnone`, `spfrcptto`)

New in JES MS 6.3-0.15. The `spfhello`, `spfmailfrom`, and `spfrcptto` channel options, when placed on a source TCP/IP channel, cause the MTA to attempt an SPF lookup at the corresponding stage of the SMTP dialogue. `spfnone`, the default, disables such SPF lookups.

With `spfhello` set (so that SPF verification of the claimed EHLO/HELO domain is attempted), possible SMTP error results (rejections) are:

```
451 4.4.3 Temporary error in SPF verification of HELO domain
500 5.5.2 Permanent error in SPF verification of HELO domain
451 4.4.3 Permanent error in SPF verification of HELO domain
500 5.5.2 Permanent error in SPF verification of HELO domain
451 4.3.0 SPF verification failed
451 4.3.0 SPF verification failed: explanation
550 5.7.1 SPF verification failed
550 5.7.1 SPF verification failed: explanation
```

The specific cases are as follows. The interpretation of the result of an SPF lookup is controlled by MTA options such as `spf_smtp_status_temperror` and `spf_smtp_status_permerror`. While temporary SPF lookup errors are normally configured to be considered as temporary errors and permanent SPF lookup errors are normally configured to be considered as permanent errors, accomplished by setting `spf_smtp_status_temperror` to 4 and `spf_smtp_status_permerror` to 5 respectively, each such option can take any of the values 2 (ignore the error condition), 4 (treat it as temporary), or 5 (treat it as permanent). Thus, with a temporary error from the SPF lookup, then the setting of the `spf_smtp_status_temperror` MTA option to 2, 4, or 5 controls, respectively, whether that SPF lookup problem is considered okay, or results in a temporary error such as (in this example, when `spfhello` is set):

```
451 4.4.3 Temporary error in SPF verification of HELO domain
```

or a permanent error such as (in this example, when `spfhello` is set):

```
500 5.5.2 Temporary error in SPF verification of HELO domain
```

Similarly, with a permanent error returned from the SPF lookup, the setting of the `spf_smtp_status_permerror` MTA option to 2, 4, or 5 controls, respectively, whether the permanent error returned by the SPF lookup is ignored (considered to be an okay condition) or results in a temporary error such as (in this example, when `spfhello` is set):

451 4.4.3 Permanent error in SPF verification of HELO domain

or a permanent error such as (in this example, when spfhelo is set):

500 5.5.2 Permanent error in SPF verification of HELO domain

New in 8.0, an SPF HELO/EHLO check failure will result in a "J" record in the [MTA message transaction log file](#), if message transaction [logging](#) has been enabled.

New in 8.0, the error text is configurable via various `error_text_spf_ehlo_*` MTA options. Also, the SMTP error codes and extended codes have been adjusted to accord with draft-ietf-appsawg-email-auth-codes-07.

With `spfmailfrom` set (so that SPF verification of the claimed MAIL FROM address is attempted), possible SMTP error results (rejections) are, in the case of temporary errors, and depending upon the setting of the `spf_smtp_status_temperror` MTA option, either:

451 4.4.3 Temporary error in SPF verification of MAIL FROM domain

or

550 5.5.2 Temporary error in SPF verification of MAIL FROM domain

In the case of permanent errors, depending upon the setting of the `spf_smtp_status_permerror` MTA option, either:

451 4.4.3 Permanent error in SPF verification of MAIL FROM domain

or

550 5.5.2 Permanent error in SPF verification of MAIL FROM domain

In the case of an SPF fail result (the lookup shows that such a MAIL FROM address is *not* authorized), depending upon the setting of the `spf_smtp_status_fail` and `spf_smtp_status_fail_all` MTA options, either

451 4.4.3 SPF verification failed

or

550 5.7.1 SPF verification failed

or when additional explanation is available, either

451 4.4.3 SPF verification failed: *explanation*

or

550 5.7.1 SPF verification failed: *explanation*

In the case of an SPF soft failure, depending upon the setting of the `spf_smtp_status_softfail` and `spf_smtp_status_softfail_all` MTA options, either:

451 4.4.3 SPF verification failed (soft)

or

550 5.7.1 SPF verification failed (soft)

With `spfrcptto` set, so that the attempt to perform an SPF verification of the MAIL FROM address is delayed until the RCPT TO stage of processing, possible errors are:

```
450 4.5.1 temporary error in SPF verification of MAIL FROM domain (domain)
550 5.5.0 temporary error in SPF verification of MAIL FROM domain (domain)
450 4.5.1 permanent error in SPF verification of MAIL FROM domain(domain)
550 5.5.0 permanent error in SPF verification of MAIL FROM domain(domain)
450 4.5.1 SPF verification of MAIL FROM domain (domain) failed
450 4.5.1 SPF verification of MAIL FROM domain (domain) failed: spf-explanation
550 5.5.0 SPF verification of MAIL FROM domain (domain) failed
550 5.5.0 SPF verification of MAIL FROM domain (domain) failed: spf-explanation
450 4.5.1 SPF verification of MAIL FROM domain (domain) failed (soft)
550 5.5.0 SPF verification of MAIL FROM domain (domain) failed (soft)
```

New in 8.0, the error text used at MAIL FROM stage (`spfmailfrom`) and RCPT TO stage (`spfrcptto`) is configurable via various `error_text_spf_*` MTA options; note that these options had existed since JES MS 6.3, but were not effective until 8.0. Also new in 8.0, the SMTP error codes and extended codes have been adjusted to accord with `draft-ietf-appsawg-email-auth-codes-07`.

Note that when SPF lookups are configured and a message is allowed in (due to either passing the SPF lookup check, or due to a configuration that allows in even messages with certain sorts of SPF lookup failures, or failure responses from SPF), then the MTA will add a "Received-SPF:" header line:

```
Received-SPF: spf-result (spf-explanation)
```

Note that SPF is prone to causing problems for autoforwarding; (such problems are *not* with the MTA's implementation, but rather are due to fundamental oversights in the original design of SPF). Use of `SRS address encoding` is one approach to work around SPF's fundamental difficulties with autoforwarding.

33.3.28.12.1 SPF_LOCAL mapping table

The `SPF_LOCAL` mapping table, if defined, provides a way to avoid performing actual DNS lookups for SPF verification of any domains matching a pattern in the mapping table: instead, the mapping table template of a matching entry will be used as if it were the DNS result of an SPF lookup. Thus, this mapping table allows providing "short-circuited" results for specified (typically local) domains.

The syntax is:

```
SPF_LOCAL
    domain-pattern    result
```

33.3.28.13 Triggering new jobs (threaddepth)

The `threaddepth` channel option tells the [Job Controller](#) when to start a new channel "job" to handle messages: for multithreaded channels, when to start a new thread (if the process is allowed to have more threads) or failing that a new process (if more processes are allowed); for single threaded channels, when to start a new process (if more processes are allowed).

For multithreaded channels, the `threaddepth` channel option controls how many messages are handled in any one thread before the channel will consider starting to use another thread.

In particular, the MTA's [SMTP client](#) (for channels not marked with the `daemon` channel option) sorts outgoing messages to different destinations to different threads. The `threaddepth` channel option may be used to instruct the MTA's 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 value specified must be greater than 1 and less than 10000. The default as of JES MS 6.0 is `threaddepth 10`. (This is a change from previous versions, in which the default was 128.)

Use of `threaddepth` may be of particular interest for achieving multithreading with [daemon router](#) on a [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.

Similarly, the `threaddepth` option affects operation of the multithreaded [ims-ms channel](#).

For single threaded channels, such as the [conversion, process, and reprocess channels](#), the `threaddepth` channel option controls how many messages are handled in a single process; more messages cause the [Job Controller](#) to create another process (up to the `maxjobs` channel option setting for the channel and the `job_limit` Job Controller option value for the `pool` in which the channel runs) to process the messages.

33.3.29 TLS and SASL channel options

A number of channel options related to SASL (SMTP AUTH) and/or TLS use.

33.3.29.1 Credentials for client SMTP AUTH use channel options: `authpassword`, `acceptvalidaddresses`, `externalidentity`

The `authusername`, `authpassword`, and `externalidentity` channel options may only be set (are only valid) in Unified Configuration; they replace the (legacy configuration

only) TCP/IP-channel-specific options [AUTH_PASSWORD](#), [AUTH_USERNAME](#), and [EXTERNAL_IDENTITY](#).

SASL authentication will be attempted if either the [maysaslclient](#) or [mustsaslclient](#) channel option is set, with success required for message transmission if [mustsaslclient](#) is set.

The PLAIN and EXTERNAL SASL mechanisms are currently supported. The [authusername](#) and [authpassword](#) channel options provide the credentials for the PLAIN mechanism and the [externalidentity](#) channel option provides the identity string for SASL EXTERNAL. ([externalidentity](#) can be set to the empty string to enable SASL EXTERNAL without an identity string.)

33.3.29.2 Authenticated originator information processing (authrewrite)

The [authrewrite](#) option may be used on a source channel to have the MTA propagate authenticated originator information, if available, into the headers. Normally the SMTP AUTH information is used (specifically, the user's canonical e-mail address, that is, the value of the `mail` attribute, found when looking up the user for authentication), though this may be overridden via the [FROM_ACCESS](#) mapping. [authrewrite](#) takes a required bit-encoded integer value as an argument, according to the following table:

Table 33.16 [authrewrite](#) option values

Bit	Value	Usage
0-3	1	Add a Sender: header line, or a Resent-sender: header line if a Resent-from: or Resent-sender: was already present, containing the AUTH originator
0-3	2	Add a Sender: header line containing the AUTH originator
0-3	3	Use the AUTH_REWRITE mapping table, probing with any Resent-Sender: and Resent-From: info if present, and otherwise probing with Sender: and From: info
0-3	4	Use the AUTH_REWRITE mapping table, probing with Sender: and From: info
0-3	5	Add a From: header line, or a Resent-From: header line if a Resent-From: or Resent-Sender: was already present, containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!
0-3	6	Add a From: header line containing the AUTH originator. This is NOT RECOMMENDED and CONTRARY TO INTERNET STANDARDS , and likely to HARM the security of your users. This option should almost NEVER be used: THIS MEANS YOU!
4	16	(New in 6.2) If set, apply the AUTH_REWRITE mapping table, even if SMTP AUTH has not been used
5	32	(New in 6.2) If set, probes to AUTH_REWRITE include the source-channel as a prefix field, separated by a vertical bar character from the rest of the probe string; that is, when this bit is set then probes take the form: <i>source-channel env-from [resent-]sender [resent-]from auth-originator</i>
6	64	(New in 7.2-7.02.) If set, use the rewritten version of the envelope from address in constructing the AUTH_REWRITE probe.
7	128	(New in 7.2-7.02.) If set, use the canonical version of the envelope from address in constructing the AUTH_REWRITE probe. Bit 6 (value 64) is a no-op if this bit is set.
8	256	(New in 7.3-11.01.) If set, add the value of the AUTH parameter from the SMTP MAIL FROM command to the AUTH_REWRITE probe, appearing just after the authorized originator address.

33.3.29.2.1 AUTH_REWRITE mapping table

Certain values of the [authrewrite](#) channel option cause the [AUTH_REWRITE](#) mapping table to be consulted to allow for more complex decision making and alterations of addresses. And bits of [authrewrite](#) also affect the form of probe to the [AUTH_REWRITE](#) mapping table.

Probes for the AUTH_REWRITE mapping table normally have the following format:

source-channel|env-from|[resent-]sender|[resent-]from|auth-originator|auth-parameter

Note that the *source-channel* field and its vertical bar suffix is only present if (new in JES MS 6.2) bit 5 (value 16) is set in the *authrewrite* argument, and *auth-parameter* and its vertical bar prefix are only present if (new in 7.3-11.01) bit 8 (value 256) is set in the *authrewrite* argument.

With *authrewrite 3*, the probes preferentially use any Resent-Sender: or Resent-From: header line values present, whereas with *authrewrite 4* the probes always use Sender: and From:. (Note that normally the AUTH_REWRITE mapping table is only consulted when a submission has included SMTP AUTH info; that is, in order for the AUTH_REWRITE mapping table to be consulted not only must the relevant incoming channel be marked with an *authrewrite* value of 3 or 4, but also the submission included use of the SMTP AUTH command. However, if bit 4 (value 16) is set in the *authrewrite* channel option's argument, then AUTH_REWRITE will be consulted even for non-authenticated submissions.)

New in 7.2-7.02, bit 6 (value 64) of *authrewrite* will, if set, cause a rewritten version of the envelope from address to be used for the *env-from* address in the probe as opposed to the original form given in the SMTP MAIL FROM command. The specific rewritten form used is controlled by bit 7 (value 128): If set the canonical form return address will be used, if clear the normally rewritten form will be used instead. These rewritten forms are useful when accessing checking is done using the AUTH_REWRITE mapping in order to prevent envelope from forgery by authenticated users.

New in 7.0.5, if bit 9 (value 512) of *authrewrite* is set, the final tag set by the *_ACCESS mappings will be prefixed to the AUTH_REWRITE mapping probe.

As of the 8.0 release, the following input flags will be set:

- \$A if SASL authentication has succeeded
- \$E if EHLO (EMSMTP) was used
- \$L if LHLO (LMTP) was used
- \$P if POP-before-SMTP was used
- \$R if this is an internal channel enqueue operation, *i.e.*, from a [conversion](#), [process](#), [reprocess](#), or similar sort of channel
- \$T if a SSL/TLS security layer has been negotiated

If the mapping table output contains a \$J, \$j, \$K, or \$k, then the envelope From address is replaced with the specified string. If the mapping table output contains a \$Y, \$y, \$T, or \$t, then a Sender: header line is added (if *authrewrite 3* was specified and if a Resent-Sender: or Resent-From: was already present, then a Resent-Sender: header line is added instead of a Sender: header line) containing the specified string.

If the mapping table output contains a \$Z or \$z, then a From: header line is added (a Resent-From: in the case of *authrewrite 3* and a Resent-From: or Resent-Sender: header line already being present) containing the specified string. (Such replacing of the From: header address is **NOT RECOMMENDED** and **CONTRARY TO INTERNET STANDARDS** and quite likely to **HARM** the overall security of your users. It should almost **NEVER** be done:

THIS MEANS YOU! Despite the wishes and mistaken notions of many sites and users, the From: header line, in Internet e-mail, is **NOT INTENDED** to represent the "real" originator of a message; it is intentionally defined permitting alternate usages.)

New in 7.3-11.01, if a \$O is specified, then another vertical-bar-separated string will be read from the mapping result string and used to set or override the value of the SMTP AUTH parameter for the current transaction. The `saslpassth` channel option may then be applied to the destination channel to cause this value to be propagated as an AUTH parameter on the SMTP MAIL FROM command.

New in JES MS 6.2, if a \$N is specified, then the message will be rejected. Optional rejection text may be specified after another vertical bar character, |. And as of JES MS 6.3, \$X may also be used to specify the extended error code (specified before the \$N text, separated by a |) in the form x.y.z. In the absence of such optional text and optional extended error code, the default text "invalid originator address used" and default extended error code 5.7.0 will be used.

When using multiple such flags, separate the string arguments with the vertical bar character, |, and specify the string arguments in the order listed in the paragraph above; that is,

```
$J$Y$Z#env-from|sender|from-header
```

or

```
$X$N|error-code|rejection-text-string
```

Technically, one could use all five flags in the same entry, though it does not seem likely to be useful:

```
$J$Y$Z$X$N#env-from|sender|from-header|error-code|rejection-text-string
```

33.3.29.3 SMTP authentication and SASL (maysasl, maysaslclient, maysaslserver, mustsasl, mustsaslclient, mustsaslserver, nosasl, nosaslclient, nosaslserver, disconnectbadauthlimit)

As of Messaging Server 7.0-3.01, `maysasl` and `mustsasl` take effect for the SMTP client direction, as well as the SMTP server direction.

The `maysasl`, `maysaslclient`, `maysaslserver`, `mustsasl`, `mustsaslclient`, `mustsaslserver`, `nosasl`, `nosaslclient`, `nosaslserver`, and `disconnectbadauthlimit` channel options are used to configure SASL use, specifically the use of the AUTH command, during the SMTP protocol by SMTP based channels such as TCP/IP channels. `nosasl` is the default, and means that SASL authentication will not be permitted nor attempted. It subsumes `nosaslserver`, which means that the SMTP server will not permit SASL authentication, and `nosaslclient`, which means that the SMTP client will not attempt SASL authentication.

Specifying `maysaslserver` will cause the SMTP server to permit clients to attempt to use SASL authentication. Specifying `mustsaslserver` will cause the SMTP server to insist that

clients use SASL authentication: the SMTP server will not accept messages unless the remote client successfully authenticates; unless authentication has been performed, the SMTP server will issue an error to any attempted MAIL FROM: command of:

```
530 5.7.0 No AUTH command has been given.
```

Note that the authentication code performs various checks on the user account when attempting to authenticate, as when a client attempts to authenticate to the MTA's SMTP server. This may result in authentication errors being returned to the SMTP server, which will in turn issue an SMTP error back in response to the SMTP AUTH attempt. Some errors of note are discussed in [Authentication errors and resultant SMTP errors](#).

New in Messaging Server 7.0 update 1 (Messaging Server 7.0-3.01) is support for limited SASL capabilities in the MTA's SMTP client. Thus it is new in Messaging Server 7.0 update 1 that the (previously existing but not meaningful) keywords `maysaslclient`, `mustsaslclient` have meaning, and the (previously existing but now with enhanced meaning) `nosaslclient`, `maysasl`, and `mustsasl` channel options truly affect SMTP client operation. SASL authentication will be attempted by the SMTP/LMTP client if the `maysaslclient`, `mustsaslclient`, `maysasl`, or `mustsasl` channel options are set---and must succeed in order for message transmission if `mustsaslclient` or `mustsasl` is set. The PLAIN and EXTERNAL SASL mechanisms are currently supported. The [AUTH_PASSWORD and AUTH_USERNAME TCP/IP-channel-specific options](#) provide the credentials for the plain mechanism and the [EXTERNAL_IDENTITY TCP/IP-channel-specific option](#) provides the identity string for SASL EXTERNAL. (EXTERNAL_IDENTITY can be set to the empty string to enable SASL EXTERNAL without an identity string.)

Normal configuration includes setting `maysaslserver` on the [tcp_local channel](#) and `mustsaslserver` on the [tcp_submit channel](#). As of Messaging Server 7.0u1, `maysaslserver` is placed also on the [tcp_intranet channel](#) definition. Additional discussion of normal configuration can also be found in [Blocking SMTP relaying](#).

New in JES MS 6.2 is the `disconnectbadauthlimit` channel option, applicable to source channels. It takes a (required) integer argument, specifying an upper limit on the number of bad (failed) SMTP AUTH attempts that will be permitted during a single SMTP session (connection). The default is 3. (Note that this default of 3 complies with the recommendation in [RFC 4954](#) that servers permit at least 3 authentication attempts prior to disconnecting due to failed attempts.) Once a client's unsuccessful SMTP AUTH attempts reaches the specified number, the SMTP server will close the connection after rejecting the SMTP AUTH attempt, including in the SMTP AUTH rejection error the additional text: "(bad authentication limit reached; disconnecting)".

See also the [saslswitchchannel](#) channel option, to cause source channel "switching" based upon successful client authentication. And see also the [sasltrustauth](#) and [saslpassthauth](#) channel options for control of the handling of any MAIL FROM AUTH parameter value. And see also the [authrewrite](#) channel option for some options on propagating SMTP AUTH information into message headers.

Note that client configuration may be required in order to get clients to make use of the MTA's SMTP AUTH support (that is, to get clients to attempt to authenticate). For instance, in order for Messenger Express (Webmail) and Communications Express (UWC) to use SMTP AUTH (SASL), one must set the `smtppauthuser` and `smtppauthpassword` MSHTTP options in Unified Configuration (or the `smtppauthuser` and `smtppauthpasswordconfigutil` http parameters in legacy configuration) to the user ID (and corresponding password) of a store

administrator (a user who exists in the list in the `admins` Message Store option in Unified Configuration, or in legacy configuration the `store.admins` list--often for instance, a user id of `admin`). (This will cause the `mshttpd` server to use the specified credentials to "vouch" for the identity of the sending user--who in turn has already had to login to `mshttpd`.)

33.3.29.4 Automatic use of AUTH EXTERNAL at MAIL FROM (explicit`saslexternal`, implicit`saslexternal`)

The SUBMIT/SMTP authentication model when authentication credentials are provided by an SSL/TLS client certification is for the SUBMIT/SMTP client to issue an AUTH EXTERNAL command after the connection is secured with SSL/TLS. Unfortunately, several popular clients do not issue an AUTH EXTERNAL command and instead rely on the binding being done automatically.

The `implicitsaslexternal` source channel option causes the SMTP/SUBMIT server to perform an implicit AUTH EXTERNAL SASL operation when a MAIL FROM command is received and the following conditions have been met:

- The `mustsaslserver` channel option at a minimum (or `mustsasl`) is in effect and no authentication operations have been performed.
- An SSL/TLS layer has been successfully negotiated.
- The client provided a valid certificate as part of the SSL/TLS exchange.

The `explicitsaslexternal` source channel option disables this behavior. It is the default.

33.3.29.5 Transport Layer Security (may`tls`, may`tlsclient`, may`tlsserver`, must`tls`, must`tlsclient`, must`tlsserver`, not`tls`, not`tlsclient`, not`tlsserver`, `tlsswitchchannel`)

The `maytls`, `maytlsclient`, `maytlsserver`, `musttls`, `musttlsclient`, `musttlsserver`, `nottls`, `nottlsclient`, `nottlsserver`, and `tlsswitchchannel` channel options are used to configure STARTTLS use during the SMTP protocol by SMTP based channels such as [TCP/IP channels](#).

Note that prior to 7.0.5, the `LMTP server` did not support TLS use; as of 7.0.5, the LMTP server does support TLS, configured via the same `maytls`, `maytlsserver`, `musttls`, `musttlsserver`, channel options used to configure SMTP server TLS support.

`nottls` is the default, and means that STARTTLS will not be permitted or attempted. It subsumes the `nottlsclient` channel option, which means that TLS use will not be attempted by the SMTP/LMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections) and the `nottlsserver` channel option, which means that TLS use will not be permitted by the SMTP/LMTP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP/LMTP 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 SMTP/LMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP/LMTP server that supports TLS, and `maytlsserver`, which means that the SMTP/LMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving

messages. Note that `maytls*` settings mean that the MTA *will* want to use TLS with remote sides that support STARTTLS, while allowing remote sides that do not have STARTTLS support to communicate without TLS; but `maytls*` settings do *not* inherently mean that the MTA will "fall back" to non-TLS use when TLS negotiation is attempted but fails: failure of TLS negotiation will result in that connection being closed as a failed connection (recorded with an "X" record). As of 8.0, with `maytlsclient` set, the MTA's client will attempt a new connection to attempt sending without TLS in cases where the remote SMTP server advertised TLS support but where the actual TLS negotiation failed; prior to 8.0, a failure in the TLS negotiation would immediately abort the delivery attempt for the message.

Specifying `musttls` will cause the MTA to insist upon TLS in both outgoing and incoming connections; e-mail will not be exchanged with remote systems that fail to successfully negotiate TLS use. It subsumes `musttlsclient`, which means that the SMTP/LMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP/LMTP servers that do not successfully negotiate TLS use (the MTA will issue the STARTTLS command and that command must succeed), and `musttlsserver`, which means that the SMTP/LMTP 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. When `musttls` or `musttlsserver` is on a channel, then unless TLS has been successfully negotiated all MAIL FROM: attempts will be rejected with the error:

```
530 5.7.0 No STARTTLS command has been given.
```

The `tlsswitchchannel` channel option is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. (This includes either successful STARTTLS use on a "regular" port, or use of the deprecated approach of negotiating upon connection to a "dedicated to TLS" port, usually port 465, configured via the Dispatcher's `ssl_ports` option in Unified Configuration, or its `TLS_PORT` option in legacy configuration.) `tlsswitchchannel` takes a required value, specifying the channel to which to switch.

Note that TLS library initialization is performed for any SMTP/LMTP channel which has any TLS usage permitted (or required). In particular, TLS library initialization will be performed by the TCP client for a channel marked merely `maytlsserver`. (This overhead is normally fairly negligible.)

Note that these options affect only TLS use negotiated at the SMTP protocol level via STARTTLS; they do not affect potential TLS use triggered by connection to a port dedicated to TLS use such as with the `ssl_ports` Dispatcher service option.

33.3.29.6 Microsoft Exchange gateway channels (`msexchange`, `nomsexchange`)

The `msexchange` channel option may be used on [TCP/IP channels](#) to tell the MTA that this is a channel that communicates with Microsoft[®] Exchange gateways and clients. Use of the option tells the MTA to try and accommodate nonstandard behavior on the part of Microsoft Exchange. Exactly what nonstandard behaviors are dealt with is subject to change.

Currently the `msexchange` channel option on a channel configured to allow TLS use (see the [*tls* channel options](#)) causes advertisement (by the MTA's SMTP server) and recognition (by the MTA's SMTP client) of the non-standard TLS capability string, in addition to the standard STARTTLS capability string, to indicate that TLS is supported.

New in 7.0.5, setting `msexchange` on a destination channel will cause the MTA, if performing any sort of MIME processing operation, to remove any `Content-disposition:` header line from any text/calendar message parts, as despite `Content-disposition:`'s long-standing existence as a standardized header line, not to mention the basic MIME rule that unrecognized `Content-*` header lines should be ignored, Microsoft® Outlook's handling of text/calendar parts is disturbed when such parts have a `Content-disposition:` specified. So specifying `msexchange` on a channel sending to Microsoft Exchange, if text/calendar parts will flow through that channel, should allow Microsoft Outlook to process calendar parts more successfully.

`nomsexchange` is the default.

33.3.29.7 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA provides support for Postfix's XCLIENT SMTP extension. The PostFix documentation for the extension can be found here:

http://www.postfix.org/XCLIENT_README.html

Use of XCLIENT is controlled by three main source channel keywords, `noxclient`, `xclient`, and `xclientsasl`, and variants `xclientrepeat` and `xclientsaslrepeat`. `noxclient` is the default, and means that XCLIENT is not advertised in the response to EHLO and the XCLIENT command itself is disabled. If `xclient` is set the XCLIENT command is enabled and the NAME, ADDR, PORT, PROTO, and HELO attributes may be used. `xclientsasl` enables the LOGIN attribute in addition to all the others. It should be noted that LOGIN specifies an external identity that must then be bound to the session identity through the use of SASL EXTERNAL.

By default, only one set of XCLIENT commands is allowed in a single SMTP session. Specifying `xclientrepeat` allows groups of XCLIENT commands to be repeated, allowing a proxy or similar agent to share a connection between multiple clients. `xclientsaslrepeat` allows multiple groups of XCLIENT commands including LOGIN. Note that care should be taken when these keywords are used since the server cannot determine the origin of a given XCLIENT command.

The primary visible effect of XCLIENT is on the contents of the `Received:` field the MTA adds. For example, if this XCLIENT command was executed:

```
xclient name=foo.domain.com addr=1.2.3.4 helo=bar.domain.com port=12345
```

it would result in a header of the general form:

```
Received: from bar.domain.com (foo.domain.com [1.2.3.4])
  by server.domain.com (Oracle Communications Messaging Server 7.0.5.32
  64bit (built Aug 18 2014)) with imapsubmit
  id <010J9P51WPFC007KNZ@server.domain.com> for user@domain.com;
  Mon, 20 Aug 2012 08:17:31 -0700 (PDT)
```

However, the ADDR and PORT attributes also change the contents of the `transportinfo` that appears in various mapping table probes, such as the probe to `PORT_ACCESS`. Given the preceding XCLIENT command, the `transportinfo` part of the mapping probes would change to something like:

```
TCP | <destaddr> | 25 | 1.2.3.4 | 12345
```

33.3.29.8 AUTH parameter handling (`saslpassauth`, `nosaslpassauth`, `sasltrustauth`, `nosasltrustauth`)

The SMTP Service Extension for Authentication, specified in [RFC 4954](#), defines an AUTH parameter for the MAIL FROM command. This parameter is normally used to pass information about the identity associated with the agent that submitted the message between SMTP servers. As required by the specification, the MTA always accepts and retains any value presented in an AUTH parameter.

If the `saslpassauth` channel option is set on a destination channel, any AUTH parameter value associated with the message will be passed on to the next SMTP server, assuming that server supports the authentication extension. `nosaslpassauth` is the default.

New in Messaging Server 7.3-11.01, if the `sasltrustauth` channel option is set on a source channel, any value presented in the AUTH parameter will be promoted to the authenticated originator address that's used throughout the MTA. `nosasltrustauth` is the default. The `sasltrustauth` option should be used with great care because the AUTH parameter is not, in general, trustworthy and the authenticated originator address is used for a variety of authentication checks. So setting `sasltrustauth`, if not done thoughtfully and carefully, may negate the value of certain authentication checks and allow more malicious spoofing of e-mail. Normally `sasltrustauth` would only be appropriate on a channel that is dedicated to receiving messages from a trustworthy (and itself careful and meticulous to require authentication) source; note that such a source must not only verify any authentication on the messages it accepts, but indeed *require* authentication on any of its incoming messages that it will relay to your host, or itself be part of a chain of such trusted relaying. The situation to avoid is trusting MAIL FROM AUTH values relayed by a -- possibly reliable enough on what it happened to authenticate itself--host that itself accepted for relaying a message with an unreliable MAIL FROM AUTH parameter.

An example of appropriate `sasltrustauth` use would be where there is a user-client-facing, dedicated-to-accepting-message-submissions host that relays to your host. Then on your host, on a channel dedicated to accepting only messages from that client-submission host, use of `sasltrustauth` could allow desirable passing along of known-to-be-accurate AUTH parameters, without opening the security door wide to passing along potentially inaccurate AUTH parameters.

See also the [AUTH_ACCESS mapping table](#), which can affect the MAIL FROM AUTH parameter.

33.3.29.9 `saslruleset` Option

RESTRICTED: Not yet implemented.

33.3.29.10 Channel switching based on SMTP authentication (`saslswitchchannel`, `nosaslswitchchannel`)

The `saslswitchchannel` channel option is used to cause incoming connections to be switched to a specified channel upon a client's successful SASL use. (See the [`maysasl*`](#) and [`mustsasl*`](#) channel options for configuration of permitting/requiring SMTP AUTH and SASL use.) `saslswitchchannel` takes a required value, specifying the channel to which to switch.

`nosaslswitchchannel` is the default, and means that channel switching is not performed upon a client's successful SASL use.

See also the `mailSMTPSubmitChannel` user LDAP attribute, (or as of the 8.0 release, whatever LDAP attribute is named by the `ldap_auth_attr_submit_channel` MTA option) which when set on a user entry will cause channel "switching" to the specified channel; it thus permits "finer-grained" channel switching than `saslswitchchannel` which merely switches all authenticated submissions to a particular named channel.

See also the (new in JES MS 6.3) `userswitchchannel` channel option which, in conjunction with site-selected user or domain LDAP attributes, also allows "fine-grained" channel switching, in this case based merely on the *purported* From: address.

The `saslswitchchannel` channel option is typically used when it is desired to distinguish between authenticated *vs.* unauthenticated submissions as a class; the `mailSMTPSubmitChannel` user LDAP attribute (or as of the 8.0 release, whatever LDAP attribute is named by the `ldap_auth_attr_submit_channel` MTA option) is typically used when it is desired to securely distinguish submissions from particular users (say to allow "special privileges" to particular users); the (new in JES MS 63) `userswitchchannel` channel option and associated LDAP attribute(s) are typically used when it is desired to make esthetic distinctions (rather than more critical "secure" distinctions) on users' submissions without requiring authenticated verification of the sender address.

See also [Blocking SMTP relaying](#) for an example of typical use of `saslswitchchannel`.

Note that any channel switching done by `saslswitchchannel` will be undone if/when a client issues a (nonstandard, new in 8.0) XUNAUTHENTICATE command. (SMTP server support for the nonstandard XUNAUTHENTICATE extension and associated XUNAUTHENTICATE command is new in 8.0; note that XUNAUTHENTICATE is not supported for the LMTP server. XUNAUTHENTICATE is only valid after successful authentication has been performed, and the capability only shows up in the EHLO response at this point at well. Successful execution of the XUNAUTHENTICATE command will return the SMTP session to an unauthenticated state.)

33.4 Header option files

Some special option files may be associated with a channel that describe how to trim the headers on messages either enqueued to, or enqueued by, that channel. This facility is completely general and may be applied to any channel; it is controlled by the [headertrim](#), [noheadertrim](#), [innertrim](#), [noinnertrim](#), [headerread](#), and [noheaderread](#) channel options.

Various MTA channels have their own channel-level option files as well. Header option files have a different format than other MTA option files and thus a header option file is always a separate file.

Note that the `test -header` utility with its `-option` switch can be used to test the effects of header trimming option files.

As of JES MS 6.3, note that the [Sieve "editheader" extension](#) provides an alternate way to alter header lines. While header trimming may be a simpler approach for performing simple changes, the Sieve "editheader" approach is more powerful in some respects such as allowing changes to specific, unrecognized-by-the-MTA, header lines, or alterations of the values on header lines.

33.4.1 Header option file location

For destination channel based header trimming to be applied upon message enqueue after normal header processing, the MTA looks in the table directory, `IMTA_TABLE:`, for header options files with names of the form `channel_headers.opt`, where `channel` is the name of the channel with which the header option file is associated. (In Unified Configuration, such header options files continue to be used.) The `headertrim` channel option and/or the `innertrim` channel option must be specified on the channel to enable the use of such a header option file.

For source channel based header trimming to be applied upon message enqueue before normal header processing, the MTA looks in the table directory, `IMTA_TABLE:`, for header options files with names of the form `channel_read_headers.opt`, where `channel` is the name of the channel with which the header option file is associated. (In Unified Configuration, such header options files continue to be used.) The `headerread` channel option must be specified on the channel to enable the use of such a header option file.

Header option files should be world readable.

33.4.2 Header option file format

Simply put, the contents of a header option file are formatted as a set of message header lines. Note, however, that the bodies of the header lines do not conform to [RFC 822](#).

The general structure of a line from a header options file is then:

```
Header-name: OPTION=VALUE, OPTION=VALUE, OPTION=VALUE, ...
```

where `Header-name` is the name of a header line that the MTA recognizes. (Any of the header lines described in this manual may be specified, plus any of the header lines standardized in [RFC 822](#), [RFC 987](#), [RFC 1049](#), [RFC 1421](#), [RFC 1422](#), [RFC 1423](#), [RFC 1424](#), [RFC 2156](#), and [RFC 2045](#). More generally, see the file `mtasdkhdr.h` in the MTA include directory.)

Header lines not recognized by the MTA are controlled by the special header line name `Other:`. A set of options to be applied to all header lines not named in the header option file can also be given on a special `Defaults:` line. Use of `Defaults:` guards against the inevitable expansion of the MTA's known header line table in future releases.

Various options may then be specified to control the retention of the corresponding header lines. The available options are:

33.4.2.1 ADD (quoted string)

The `ADD` option creates a completely new header line of the given type. The new header line contains the specified string. The header line created by `ADD` will appear after any existing header lines of the same type. The `ADD` option cannot be used in conjunction with the `Defaults:` header line type; it will be ignored if it is specified as part of an `Other:` option list.

33.4.2.2 FILL (quoted string)

The `FILL` option creates a completely new header line of the given type if and only if there are no existing header lines of the same type. The new header line contains the specified string. The `FILL` option cannot be used in conjunction with the header line type; it will be ignored if it is specified as part of an `Other:` option list.

33.4.2.3 FOLDITEMS (integer)

This option takes an integer that specifies the maximum number of "items" that can appear on a line before folding. "Items" are normally defined as comma-separated sets of tokens, but if FOLDITEMS is set to a negative value, then encoded words are also considered to be "items".

This option can be useful, for instance, when dealing with Netscape 4.7* clients which have a bug whereby they will insert an extra space between successive encoded words, unless there is a CRLF between the encoded word items.

33.4.2.4 GROUP (integer 0 or 1)

This option controls grouping of header lines of the same type at a particular precedence level. A GROUP value of 0 is the default, and indicates that all header lines of a particular type should appear together. A value of 1 indicates that only one header line of the respective type should be output and the scan over all header lines at the associated level should resume, leaving any header lines of the same type unprocessed. Once the scan is complete it is then repeated in order to pick up any remaining header lines. This header option is primarily intended to accomodate Privacy Enhanced Mail (PEM) header processing.

33.4.2.5 LINELENGTH (integer)

This option controls the length at which to fold headers. See also the discussion of the [headerlinelength](#) channel option.

33.4.2.6 MAXCHARS (integer)

This option controls the maximum number of characters which may appear in a single header line of the specified type. Any header line exceeding that length is truncated to a length of MAXCHARS. This option pays no attention to the syntax of the header line and should never be applied to header lines containing addresses and other sorts of structured information. The length of structured header lines should be controlled with the [maxheaderchars](#) and [maxheaderaddrs](#) channel options.

33.4.2.7 MAXIMUM (integer)

This option controls the maximum number of header lines of this type that may appear. This has no effect on the number of lines, after wrapping, each individual header line might consume. A value of -1 is interpreted as a request to suppress this header line type completely.

33.4.2.8 MAXLINES (integer)

This option controls the maximum number of lines all header lines of a given type may occupy. It complements the MAXIMUM option in that it pays no attention to how many header lines are involved, only to how many lines of text they collectively occupy. As with the MAXIMUM option, headers are trimmed from the bottom to meet the specified requirement.

33.4.2.9 PRECEDENCE (integer)

This option controls the order in which header lines are output. All header lines have a default precedence of zero. The smaller the value, the higher the precedence. Thus, positive PRECEDENCE values will push header lines towards the bottom of the header while negative

values will push them towards the top. Equal precedence ties are broken using the MTA's internal rules for header line output ordering.

Note that prior to JES MS 6.3, the MTA's header line precedence processing was always applied. As of JES MS 6.3, the MTA by default does not do header line re-ordering, and so header line re-ordering only takes place if header trimming is enabled. (This change was made to improve interoperability with poorly designed message signing mechanisms.)

33.4.2.10 RELABEL (header name)

This option changes a header line to another header line; that is, the name of the header is changed, but the value remains the same. For instance,

```
X-MSMail-Priority: RELABEL="Priority"  
X-Priority: RELABEL="Importance"
```

Chapter 34 Rewrite rules

34.1	The rewrite group	34-2
34.2	Application of rewrite rules to addresses	34-2
34.2.1	Rewriting: extraction of the first host or domain specification	34-3
34.2.2	Rewriting: scanning for a domain match	34-5
34.2.3	Rewriting: applying the rewrite rule template	34-7
34.2.4	Rewriting: finishing the rewriting process	34-7
34.2.5	Rewriting: rewrite rule failure	34-8
34.2.6	Syntax checks after rewriting	34-8
34.2.7	Rewriting: domain literals	34-8
34.3	Rewrite rule patterns and tags	34-9
34.3.1	Initial match-all rule	34-11
34.3.2	A rule to match percent hacks	34-11
34.3.3	A rule to match bang-style addresses	34-11
34.3.4	A rule to match any domain literal	34-12
34.3.5	Rules to match domains containing exact numbers of components	34-12
34.3.6	A rule to match any address	34-12
34.3.7	Tagged rewrite rule sets	34-13
34.4	Rewrite rule templates	34-14
34.4.1	Rewrite rule template formats	34-14
34.4.2	Rewrite rule template substitutions and control sequences	34-16
34.5	Domain database	34-34

Domain rewriting rules, or, as they are more frequently called, *rewrite rules*, play two important roles for the MTA: rewrite rules are used to convert addresses into true domain addresses, and to determine their corresponding channels. These rules are used to rewrite addresses appearing in both the transport layer and the message header. The transport layer is the message's "envelope", which contains routing information and is invisible to the user. The determination of to which [channels](#) a message should be enqueued results from rewriting its envelope To addresses.

The rewrite rules and the table of channels cooperate to determine the disposition of each address. Each address detected in a message is rewritten, starting^a with the envelope To address(es). The result of the rewrite process is a rewritten address and a "routing system" (as determined from rewriting the envelope To address); *i.e.*, the system to which the message is to be sent. Depending upon the topology of the network, the routing system may only be the first step along the path the message takes to reach its destination or it may be the final destination system itself.

After the rewrite process has finished, a search is made for the routing system among the MTA's [channels](#). Each channel will have [one](#) or [more host names](#) associated with it. The routing system name is compared against each of these names to determine to which channel to enqueue the message.

Note that the MTA provides (many) other means of manipulating addresses for the purposes of changing them for varied purposes, such as cosmetic changes, message forwarding, mailing list processing, *etc.* See for instance [Aliases](#). Rewrite rules are appropriate for global, unconditional transformations of domain names to be controlled purely by the MTA (as opposed to provisioning of domain name handling in, for instance, an LDAP directory), and are required for configuring MTA routing of messages based on envelope To address.

Every rewrite rule consists of two parts: a [pattern](#) (left hand side) followed by an equivalence string or [template](#) (right hand side). The two parts must be separated by one or more spaces. Spaces are not allowed in the parts themselves. In general, the template specifies a mailbox name (*e.g.*, username), a host/domain specification, and the name of a system attached to an existing MTA channel to which messages to this address should be enqueued. The total length of a line in the configuration file is limited to 1024 characters; the pattern is limited to 256 characters, and template (prior to substitutions) is also limited to 256 characters.

In legacy configuration, note that each rewrite rule would appear on a single line in the upper half of the MTA configuration file. Comment lines (lines beginning with a [comment character](#) such as exclamation point in the first column) but *not* blank lines could be placed between rules. Rewrite rules could also, optionally, be stored in an auxiliary database called the domain database.

In Unified Configuration, rewrite rules are stored under the [rewrite](#) XML element. But they are most conveniently viewed and edited "as if" they were in the legacy configuration `imta.cnf` file, by using the `msconfig` command `EDIT REWRITES`.

The syntax of rewrite rules is discussed in further detail in [Rewrite rule patterns and tags](#) and [Rewrite rule templates](#). First, however, [Application of rewrite rules to addresses](#) gives an overview of the action of rewrite rules in operation.

^aTechnically, rewriting begins with a preliminary rewrite of the envelope From address, for access control and source channel determination purposes. After that, the envelope To address is rewritten (possibly sensitive to the source channel), and then with destination channel(s) determined (due to the rewriting of the envelope To address) the envelope From address receives another, "real" rewriting now that the destination channels are known.

34.1 The rewrite group

In Unified Configuration, the `rewrite` group is not an option itself, but rather a list of all the MTA's [rewrite rules](#). For instance:

```
msconfig> show rewrite *
role.rewrite.rule = $* $A$E$F$U%$H$V$H@&/IMTA_HOST/
role.rewrite.rule = &/IMTA_HOST/ $U%$D@&/IMTA_HOST/
role.rewrite.rule = &/IMTA_DEFAULTDOMAIN/ $U%$D@&/IMTA_HOST/
role.rewrite.rule = .ims-ms-daemon $U%$H.ims-ms-daemon@ims-ms-daemon
role.rewrite.rule = .pipe-daemon $U%$H.pipe-daemon@pipe-daemon
role.rewrite.rule = . $U%$H$, $H@TCP-DAEMON
role.rewrite.rule = [ ] $E$R${INTERNAL_IP, $L}$U%[$L]@tcp_intranet-daemon
role.rewrite.rule = deleted-daemon $U%$H@deleted-daemon
role.rewrite.rule = .deleted-daemon $U%$H@deleted-daemon
role.rewrite.rule = inactive-daemon $U%$H@inactive-daemon
role.rewrite.rule = .inactive-daemon $U%$H@inactive-daemon
role.rewrite.rule = hold-daemon $U%$H@hold-daemon
role.rewrite.rule = .hold-daemon $U%$H@hold-daemon
```

Rewrite rules are typically most conveniently manipulated by using the `msconfig` command `EDIT REWRITES`, which allows viewing them and editing them "as if" they were the upper half of a legacy `imta.cnf` file.

34.2 Application of rewrite rules to addresses

This section presents a discussion of the operation of domain rewriting rules: how an address is parsed and then transformed via rewrite rules. This section touches briefly on the syntax of rewrite rules as such syntax relates to example addresses, but for full details on rewrite rule syntax, see [Rewrite rule patterns and tags](#) and [Rewrite rule templates](#).

There are four steps in the application of the domain rewriting rules to a given address:

1. [The first host or domain specification is extracted from the address](#). (Note that an address may specify more than one host or domain name as is the case with the address `jdoo%host1@domain.com`.)
2. After extracting the first host or domain name specification, the rewrite rules are [scanned for a matching rewrite rule](#). That is, a search is conducted for a rewrite rule whose pattern portion matches the extracted host/domain name.
3. Once a matching rewrite rule is found, the address is rewritten [according to the template portion](#) of that rule. The template also specifies the name of a routing system to which messages to this address should be routed.⁸
4. [The routing system name is then compared with the host names associated with each channel](#). If a match is found, then the message is enqueued to that channel; otherwise, the [rewriting process is considered to have failed](#). If the matching channel is the local channel, then some additional rewriting of the address may occur.

These four steps are described in detail in the following subsections. There are also special template formats which allow for variations in these four steps.

Note⁸The term "routing system" can be misleading. It does not necessarily mean the name of a system through which the message will be routed but rather is a host name, possibly fictitious, associated with a specific channel.

34.2.1 Rewriting: extraction of the first host or domain specification

The process of rewriting an address starts by extracting the first host/domain specification from the address. (Readers who are not familiar with [RFC 822](#) address conventions are advised to read that standard, at least in a cursory fashion, at this point in order 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 singleton percent sign.
4. Hosts appearing to the left of the first exclamation point.

The order of the last two items are switched if the [bangoverpercent](#) channel option is in effect on the channel that is doing the address rewriting, that is, if the channel which is attempting to enqueue the message is itself marked with the [bangoverpercent](#) channel option.⁹

Some highly hypothetical examples of addresses and the host name that would be extracted first are shown below:

Address	First host/domain specification	Comments
user@a	a	a is a "short-form" domain name
user@a.b.c	a.b.c	a.b.c is a "fully-qualified" domain name (FQDN)
user@[0.1.2.3]	[0.1.2.3]	[0.1.2.3] is a "domain literal"
@a:user@b.c.d	a	This is 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 non-standard form of routing is called a "percent hack"
user%A%B%C@D	D	A built up percent hack
user%A	A	
user%A%B	B	
user%%A%B	B	
user%A%%B	A%%B	Of questionable value
@A:user%B@C	A	
A!user	A	"Bang-style" addressing; commonly used for UUCP
A!user@B	B	
A!user%B@C	C	
A!user%B	B	nobangoverpercent channel option active; the default
A!user%B	A	bangoverpercent channel option active
@A:B!user@C	A	
@A,@B:C!user%D@E	A	Too grotesque to consider, really

Note that [RFC 822](#) does not say anything about the interpretation of exclamation points, !, and percent signs, %, in addresses. It is customary to interpret percent signs in the same manner as at signs, @, if no at sign is present, so this convention is adopted by the MTA.

The special interpretation of repeated percent signs is used to allow percent signs as part of local usernames, which is used in handling PSIMail and other 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 MTA.

The order of interpretation of exclamation points *vs.* percent signs 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. Note that the default is more "standard", although the alternate setting may be useful under some circumstances.

⁹ For instance, if this is a message being submitted to the SMTP port from a "local" client, then the enqueueing channel is typically `tcp_intranet`, or `tcp_auth` if the user authenticates. If this is a message being submitted to the SMTP SUBMIT port from a "local" client, then the enqueueing channel is typically `tcp_submit`. If it is a message coming in from a remote Internet user, then usually it will be the `tcp_local` channel doing the enqueueing.

34.2.2 Rewriting: scanning for a domain match

Once the first host/domain specification has been extracted from the address, the MTA consults the rewrite rules to find out what to do with it. Initially, the exact host/domain specification is compared with the `pattern` part of each rule (*i.e.*, the left-hand side of each rule). This comparison is (and the rest of the comparisons discussed below are also) case insensitive. Case insensitivity is mandated by RFC 822, UUCP addresses notwithstanding. The MTA is insensitive to case but preserves it whenever possible.

If the pattern matches, then the host/domain is transformed as specified by the `rewrite rule template` (right hand side). If that (transformed, as appropriate) host/domain specification then matches a channel `official host name`, the rewriting is considered complete and the address is considered to match that channel.

New in Comms Suite 7.0 is support for attempting an initial, special rewrite of a host/domain with a trailing dot. (Such a trailing dot on a host/domain name is illegal in Internet domain names, but has been tolerated in *some* contexts by the MTA for a long time. RFC 1123 points out that trailing dots are syntactically illegal in email but notes that some convention needs to exist in user interfaces where short form names can be used. Accordingly, it may be handy in contexts like SMTP submission of messages, SMTP SUBMIT, to be able to accept addresses with trailing dots, and then remove the dot while attaching special semantics to its initial presence.) New in 7.0, the MTA will attempt to rewrite the host/domain with the trailing dot present; if that fails, then the MTA will remove the trailing dot from the host/domain and then continue rewriting attempts, as normal, from that point on with the trailing dot removed.

If the host/domain specification does not match any pattern, in which case it is said to "not match any rule", then the first part of the host/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/domain specification, but only in the regular rewrite rules (those in the `imta.cnf` file or in Unified Configuration `rewrite group` -- 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, as long as there is still at least one portion of the original host/domain specification remaining. All probes that contain asterisks are only done in the regular rewrite rules table (those rewrite rules in `imta.cnf` or in the `rewrite group` in Unified Configuration); the domain database is not checked. This process proceeds until either a match is found or the host/domain specification up to (but not including) its last portion 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.

If the entire host/domain specification with the exception of its last (right-most) portion has been exhausted looking for a rewrite rule pattern that matches and a match has still not been found, then an additional check of the entire host/domain specification is performed; namely the channel host table is scanned for a matching host name associated with a channel; that is,

the entire host/domain specification is compared against any channel [official host names](#) and [host name aliases](#) on channel definitions to look for a match.

If a match has still not been found, then a special all-components-replaced-by-asterisks attempt is made, (in particular, in the case of short-form host names an * lookup is attempted) in the regular rewrite rules (those in `imta.cnf` or in the [rewrite group](#) in Unified Configuration), see [Rules to match domains containing exact numbers of components](#); and if that did not succeed then the special "match-all" rule described in [A rule to match any address](#) is attempted.

A somewhat more algorithmic view of this matching procedure is given below.

1. The host/domain specification is used as the initial value for the comparison strings `spec_1` and `spec_2`. *E.g.*, `spec_1 = spec_2 = a.b.c`.
2. The comparison string `spec_1` is compared with the pattern part of each rewrite rule in the `imta.cnf` configuration file (legacy configuration) or the [rewrite group](#) (Unified Configuration), and then the domain database, until a match is found. The matching procedure is exited if a match is found.
3. If no match is found then the leftmost, non-asterisk part of `spec_2` is converted to an asterisk. *E.g.*, 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`; *etc.* The resulting comparison string `spec_2` is compared with *only* the configuration file (legacy configuration) or the [rewrite group](#) (Unified Configuration). The domain database is not consulted. The matching procedure is exited if a match is found.
4. If no match is found then the first part, including any leading period, of the comparison string `spec_1` is removed. In the case where `spec_1` has only one part (*e.g.*, `.c` or `c`), the string is replaced with a single period, `."`. If the resulting string `spec_1` is of non-zero length, then we return to Step 1. If the resulting string has zero length (*i.e.*, was previously `."`) then the lookup process has failed and we exit the matching procedure.

For example, suppose the address `dan@sc.cs.cmu.edu` is to be rewritten. This causes the rewriter to look for the following patterns in the given order:

Pattern	Files Scanned
<code>\$*</code>	configuration file/ rewrite group rules and then domain database; this is the Initial match-all rule
<code>sc.cs.cmu.edu</code>	configuration file/ rewrite group rules and then domain database
<code>*.cs.cmu.edu</code>	configuration file/ rewrite group rules only
<code>.cs.cmu.edu</code>	configuration file/ rewrite group rules and then domain database
<code>*.*.cmu.edu</code>	configuration file/ rewrite group rules only
<code>.cmu.edu</code>	configuration file/ rewrite group rules and then domain database
<code>*.*.edu</code>	configuration file/ rewrite group rules only
<code>.edu</code>	configuration file/ rewrite group rules and then domain database
<code>sc.cs.cmu.edu</code>	channel host name table (those official host names and host name aliases specified in channel definitions)
<code>*.*.*</code>	configuration file/ rewrite group rules only; this is an example of Rules to match domains containing exact numbers of components
<code>.</code>	match-all rule described in A rule to match any address

Note: Always remember that patterns involving asterisks (except the initial match-all pattern, `$*`) are only searched for in the configuration file's set of rewrite rules; no searching is done for these patterns in the domain database.

34.2.3 Rewriting: applying the rewrite rule template

Once a [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 username for the address,
2. a new host/domain specification for the address, and
3. the name of a system attached to an existing MTA channel (the "routing system") to which messages to this address should actually be sent.

Template format is discussed in detail in [Rewrite_rule_templates](#). As a quick overview, note that the most [common format for templates is \$A\%B\%C\$](#) , where A is the new username, B is the new host/domain specification, and C is the routing system. And the [format \$A\%C\$](#) (which is an abbreviation for $A\%C\%C$) is also commonly used.

[Substitution strings](#) are allowed in the template. For instance, to mention some of the more commonly used substitution strings, any occurrences of `$U` in the template are replaced with the username from the original address, any occurrences of `$H` are replaced with the portion of the host/domain specification that was *not* matched by the rule, and any occurrences of `$D` are replaced by the portion of the host/domain specification that *was* matched by the rewrite rule. [Summary of template substitutions and control sequences](#) contains a summary of these and other substitution strings which are presented in detail in [Rewrite rule template substitutions and control sequences](#).

As an example, suppose that the host/domain specification `jdoh@domain.com` has matched the rewrite rule

```
domain.com          $U@DOMAIN.COM
```

Then the template will produce the username `jdoh`, the host/domain specification `DOMAIN.COM`, and the routing system `DOMAIN.COM`. In a slightly more complicated example, assume that the host/domain specification has matched the rewrite rule

```
.com                $U%$H%D@TCP-DAEMON
```

In this case, `$U` = `jdoh`, `$H` = `domain`, and `$D` = `.com`. The template produces the username `jdoh`, the host/domain specification `domain.com`, and the routing system `TCP-DAEMON`.

34.2.4 Rewriting: finishing the rewriting process

One of two things can happen once the host/domain specification is rewritten. If the routing system is not associated with either the local channel or a channel explicitly marked with the [`routelocal`](#) channel option, or in any case when there are no additional host/domain specifications in the address, then the rewritten specification is substituted into the address replacing the original specification that was extracted for rewriting, and the rewriting process terminates.

If the routing system matches the local channel (or a channel marked with the `routelocal` channel option) and there are additional host/domain specifications that appear in the address, then 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 the MTA's way of providing support for source routing. In effect, superfluous routes through the "local system" are removed from addresses by this process.

34.2.5 Rewriting: rewrite rule failure

If a host/domain specification fails to match any rewrite rule and no default rule (that is, `match-all rule`) is present, the MTA simply uses the specification "as-is"; *i.e.*, 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. This relaxed interpretation of rewrite rule failures allows isolated MTA sites that only communicate with a small number of systems to get by without any rewrite rules whatsoever.

34.2.6 Syntax checks after rewriting

No additional syntax checking is done after the rewrite rules have been applied to an address. This laxity 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 configuration mistakes in the rewrite rules may result in messages leaving the MTA system with incorrect or illegal addresses.

34.2.7 Rewriting: 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.^a The rightmost 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/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/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 `"."`.

When domain literals are combined with domain names the number of lookup attempts gets to be quite large. This is *not* normal usage and its use is *strongly discouraged*. For example, the address `dan@[1.2].a.[3.4].b` would generate requests for:

```

[1.2].a.[3.4].b
[1.].a.[3.4].b
[].a.[3.4].b
[*.*].a.[3.4].b
.a.[3.4].b
[*.*].*. [3.4].b
.[3.4].b
[*.*].*.[3.].b
.[3.].b
[*.*].*.[].b
.[].b
[*.*].*.[*.*].b
.b
[*.*].*.[*.*].*
•

```

New in MS 7.0, the MTA supports the [RFC 2822](#) definition of spaces in domain literals as FWP, or in other words, semantically null; (note that the meaning of such spaces had not been specified in [RFC 822](#)). That is, as of MS 7.0, the MTA will canonicalize `user@[a . b . c . d]` as `user@[a.b.c.d]`, which would not have occurred in previous versions.

^a Note that the support of numeric domain literals is not required by either the MTA or [RFC 822](#). Their support is enabled by including IP literal rewrite rules in the MTA configuration.

34.3 Rewrite rule patterns and tags

Most rewrite rule patterns consist either of a specific host name that will match only and exactly that host, *e.g.*,

```
host.domain.com
```

or consist of a subdomain pattern that will match any host/domain in the entire subdomain, *e.g.*,

```
.domain.com
```

A rewrite rule pattern such as the above would match any `host.domain.com` or `host.subnet.domain.com` sort of host/domain name. Note, however, that it will *not* match the exact host name `domain.com`; to match the exact host name `domain.com`, a separate `domain.com` pattern would be needed.

The matching in rewrite rule patterns is case-insensitive: uppercase and lowercase are not significant, in either the pattern, or in the domain of the address being rewritten. (Note, however, that rewrite rule templates preserve case.)

Since as discussed in [Rewriting: scanning for a domain match](#) 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 instance, if the rewrite rule patterns

```
hosta.subnet.domain.com
.subnet.domain.com
.domain.com
```

are present in the configuration file (legacy configuration) or the [rewrite group](#) (Unified Configuration),^b then an address of `jdoue@hosta.subnet.domain.com` will match the specific `hosta.subnet.domain.com` rewrite rule pattern, while an address of `jdoue@hostb.subnet.domain.com` will match the more general `.subnet.domain.com` rewrite rule pattern, and an address of `jdoue@hostc.domain.com` will match the `.domain.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 subdomains into their configuration: in older configuration, from the file `internet.rules` stored in the MTA's table (config) directory, or in newer configurations from the `tlds.txt` file.

In older configurations, the incorporation of rewrite rules from `internet.rules` such as

```
! Ascension Island
.AC                               $U%$H$D@TCP-DAEMON
. [text. removed for.          brevity]
! Zimbabwe
.ZW                               $U%$H$D@TCP-DAEMON
```

with rewrite rule patterns that match the top level Internet domains and rewrite rule templates that rewrite addresses matching such patterns to an outgoing TCP/IP channel, 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 out an outgoing TCP/IP channel.

In newer configurations, a similar effect is obtained via the special rewrite rule:

```
. $U%$H$, $H@TCP-DAEMON
```

where the special `$, $H` sequence causes a lookup of the entire unmatched domain (the `$H`) in the `tlds.txt` file (the `$,`) listing current Top Level Domains.

IP domain literals follow a similar hierarchical matching pattern, though with right-to-left (rather than left-to-right) matching. For instance, the pattern

```
[1.2.3.4]
```

matches only and exactly the IP literal `[1.2.3.4]`, while

```
[1.2.3.]
```

matches anything in the `1.2.3.0` subnet.

In addition to the more common sorts of host or subdomain rewrite rule patterns discussed above, rewrite rules may also make use of several special patterns, summarized in [Summary of special patterns for rewrite rules](#), and discussed in the following subsections.

Table 34.1 Summary of special patterns for rewrite rules

Pattern	Name	Usage
\$*	Match-first rule	Initial, prior-to-all-else rewrite rule, matches everything
\$%	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	Match any IP domain literal.
*, **, ***, etc.	Match-exactly-n-components-domain-names rules	A rule with all asterisks will match any domain name containing that exact number of components; in particular, * matches any short form host.
.	Match-all rule	Matches any host/domain specification.

In addition to these special patterns, the MTA 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; see [Tagged rewrite rule sets](#).

34.3.1 Initial match-all rule

The special pattern \$* is applied before any other rewriting, and matches all addresses. Its usual use is as a fundamental part of a ["direct LDAP" setup](#), to achieve LDAP-based routing of domains. That is, its usual use is in conjunction with a template looking up domains in LDAP, so that domains are looked up in LDAP prior to any other rewriting. For example:

```
$*    $A$E$F$U$H$V$H@official-host-name-of-l-channel
```

34.3.2 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. This extra rule is the percent hack rule. The pattern is \$%. The pattern never changes. 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.

34.3.3 A rule to match bang-style 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 [match-all rule](#)).

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.

34.3.4 A rule to match any domain literal

If the MTA tries to rewrite an address of the form [n.m.p.q] and fails, it tries one extra rule (prior to trying [*. *.*.*], and then the "." [match-all rule](#)). This extra rule is the IP literal match-all rule. The pattern is []. The pattern never changes. This rule is only activated when more specific probes including all or part of the IP address have not matched. (See [Handling of domain literals](#) for a discussion of the order in which portions of an IP address are checked for a match.)

In particular, a rewrite rule using the [] pattern and with `ER` in the template (thus meaning that the rewrite rule applies only to envelope From addresses) is typically used in modern MTA configurations to perform a lookup of a pseudo-address constructed from the incoming source IP of SMTP connections against the [INTERNAL_IP mapping table](#) and then to "switch" the incoming SMTP connection to an "internal" channel such as `tcp_intranet`, if appropriate.

34.3.5 Rules to match domains containing exact numbers of components

Special patterns of the form `*`, `*.*`, `*.*.*`, *etc.*, may be used to provide penultimate, fall-through matches for domains containing the specified number of components (the same number of components in the original host/domain specification as the number of asterisks in the pattern). A rewrite rule pattern with the same number of asterisks as components in the original host/domain specification will be checked after the comparison of the original, unmatched host/domain specification against the channel host name table, and before the "." [match-all rule](#). That is, when no more specific rewrite rule match has been found, nor has a match been found in the channel host name table, then all components of the original host/domain specification are replaced by asterisks for one last probe (prior to the "." [match-all rule](#)). If a rewrite rule pattern containing that exact same number of asterisk components is found, then that rewrite rule is considered to match the host/domain and is applied.

One of the more common uses of this form of pattern is that of a `*` pattern for matching all short form host names (host names unadorned by any higher-level domain components). For example:

```
*          $U%$H.domain.com
```

(Though in the modern Internet environment, in the interests of encouraging proper use of correct domain names, it is often better to instead discourage all use/acceptance of short form names.)

Note that, as with all asterisk-in-place-of-component(s) probes, these probes are only made to the configuration file (legacy configuration) or [rewrite group](#) (Unified Configuration), not to the domain database.

34.3.6 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.

In times past, with a slower changing set of Top Level Domains, the "." rewrite rule was less commonly used. Instead, in the past, use of an `internet.rules` file of rewrite rules for matching the (seldom changing) known top-level Internet Domains permitted immediate feedback on addresses with clearly invalid Top Level Domains -- immediate feedback on obvious misspellings of TLDs. With that approach, "known" Internet TLDs could be routed by the rewrite rules in `internet.rules`, but "bogus" TLDs, not matching any rewrite rule, would be detected during rewriting, and rejected. As such, in the past, the special "." rule tended to be used only when the MTA did not have complete routing information available and had to defer judgment of address validity to another system or systems. In those cases, the "." pattern was used simply the MTA configuration at the expense of allowing propagation of possibly bogus addresses.

However, nowadays "." is routinely used in an important rewrite rule making a comparison against a (frequently updated) `tlds.txt` list of Top Level Domains to achieve routing of Internet addresses with apparently valid TLDs, while not propagating addresses with invalid TLDs. Nowadays, the `internet.rules` file, instead of containing distinct rewrite rules for each Top Level Domain, merely contains the one special rewrite rule:

```
.      $U%$H$, $H@TCP-DAEMON
```

Here the "." pattern causes all domain names not matched by other, more specific rewrite rules to get matched. However, the \$, compares the top-level portion of the domain substituted by \$H (in the case of this rewrite rule with a "." match, the \$H substitutes back the entire domain in the address) against the list in `tlds.txt` and the rewrite rule will only succeed if a match is found: this rewrite rule will only succeed if the top-level portion of the domain of the address being rewritten can be found in `tlds.txt`.

Note: 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!

34.3.7 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 (legacy configuration) or [rewrite group](#) (Unified Configuration), 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.

See Section 2.2.6.19 for an example of using tagged rewrite rules.

34.4 Rewrite rule templates

Once a [host/domain specification matches a rewrite rule](#), it is rewritten using the template part (right hand side) of the rule. The template specifies three things:

1. a new username for the address,
2. a new host/domain specification for the address, and
3. the name of a system attached to an existing MTA channel (the "routing system") to which messages to this address should actually be sent.

There are several general formats for rewrite rule templates, which will be discussed in [Rewrite rule template formats](#), depending upon whether an address is merely being changed or whether source routing should be explicitly specified (or even added to the address). Various [substitutions and control sequences](#) are available to further fine-tune the application of rewrite rules.

Note that the character case in templates is preserved. This is necessary when using rewrite rules to provide an interface to a mail system such as UUCP which is sensitive to character case. [Substitution sequences](#) like \$U and \$D that substitute material extracted from addresses also preserve the original case of characters. (Special [Rewrite case control substitutions](#) may be used when it is desirable to alter case, rather than preserve it.)

34.4.1 Rewrite rule template formats

A summary of the template formats for rewrite rules is presented in [Summary of template formats for rewrite rules](#). The substitution strings and control sequences which may be used with templates are discussed in [Rewrite rule template substitutions and control sequences](#).

Table 34.2 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
A@B	Treated as A%B@B
A%B@C	A becomes the new user/mailbox name, B becomes the new host/domain specification, route to C
A@B@C	Treated as A@B@C@C.
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 D

Other formats, such as A%B%C and so forth, are reserved for the implementation of future capabilities in the MTA and should not be used as their function may change in a future release.

34.4.1.1 Ordinary rewriting templates, A@B or A%B@C

The most commonly used form of rewrite rule template is A%B@C, where A is the new username, B is the new host/domain specification, and C is the routing system (official channel

name). If B and C are identical, %B may be omitted; *i.e.*, you may simply use A@C when B and C are identical.

For instance, if an MTA for domain.com has the system (host) name `host.domain.com` (the official host name of the "I" channel), and if the default email domain name is `mail.domain.com`, then typical rewrite rules could include:

```
host.domain.com    $U%host.domain.com@hostname.domain.com
mail.domain.com   $U%mail.domain.com@hostname.domain.com
```

meaning that addresses with domain names of `host.domain.com` or `mail.domain.com` will be routed, (without any beforehand domain name change) to the "I" channel.

Note that in a current Unified Configuration, such rewrite rules would typically be automatically generated, and make use of the `$D` and `&/IMTA_HOST/` and `&/IMTA_DEFAULTDOMAIN/` [substitutions](#), to appear as:

```
msconfig> show rewrite
...other rewrite rules...
role.rewrite.rule = &/IMTA_HOST/ $U%$D&/IMTA_HOST/
role.rewrite.rule = &/IMTA_DEFAULTDOMAIN/ $U%$D&/IMTA_HOST/
...other rewrite rules
```

34.4.1.2 Repeated rewriting template, A%B

The special rewrite rule template format A%B is used for "meta-rules" that require additional rewriting after their application. When an A%B pattern is encountered, A becomes the new username and B becomes the new host/domain specification, and then 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.

For example, the rule

```
.removeable      $U%$H
```

has the effect of removing all occurrences of the `.removeable` domain from the ends of addresses.

Extreme care must be taken when using these repeating rules; careless use can create a "rules loop" that will hang the MTA in an infinite loop. (The MTA will attempt to detect simple cases of such loops to abort out after 100 repetitions: but even so this is (a) inefficient use of the MTA computation time, and (b) not fail-safe, as more complex loops may not be detectable by the MTA.) For this reason meta-rules should only be used when absolutely necessary. Be sure to test them with the command `imsimta test -rewrite`.

34.4.1.3 Specified route rewriting templates, A@B@C or A@B@C@D

The special rewrite rule template format A@B@C works in the same way as the [usual A%B@C rule](#), except that the routing system C will also be inserted into the address as a source route. This inclusion of the routing system in the address may be needed by some channels that have

to establish a connection to the routing system and determine the name of the routing system from the envelope To address. For instance, the rewrite rule

```
host1.domain.com      $U@host1.domain.com@hub.domain.com
```

would rewrite the address `jdoh@host1.domain.com` into the source routed address `@hub.domain.com:jdoh@host1.domain.com`. The routing system will be `hub.domain.com`.

The template format `A@B@C@D` uses `A` as the new username, `B` is the new host/domain specification, `C` is inserted as a source route, and `D` is the routing system. This is the most general template format available.

34.4.2 Rewrite rule template substitutions and control sequences

Substitutions are used to substitute into the rewritten address a character string the value of which is determined by the particular substitution sequence used. For instance in the template

```
$U@domain.com
```

the `$U` is a substitution sequence. It causes the username portion of the address being rewritten to be substituted into the output of the template. Thus, if `jdoh@host1.domain.com` was being rewritten by this template, the resulting output would be `jdoh@domain.com`, the `$U` substituting in the username portion, `jdoh`, of the original address.

Special control sequences may also appear in rewrite rule templates. These 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/domain specification being examined, but other aspects of the address being rewritten must meet conditions set by the control sequence or sequences. For instance, the `$E` control sequence requires that the address being rewritten be an envelope address while the `$F` sequence requires that it be a forward pointing address. Thus, the rewrite rule

```
domain.com      $U@domain.com$E$F
```

will only apply to (*i.e.*, only rewrite) envelope To addresses of the form `user@domain.com`. If a domain/host specification matches the pattern portion of a rewrite rule but doesn't meet all of the criteria imposed by control sequences in the rule's template, then the rewrite rule fails and the rewriter continues to look for other applicable rules. This makes possible sets of rewrite rules such as

```
domain.com      $U%domain.com@conversion-daemon$Nconversion  
domain.com      $U@domain.com
```

which will result in messages to `user@domain.com` being passed to the conversion channel. However, should the conversion channel rewrite a message with the address `user@domain.com`, that message will not again pass through the conversion channel. This then allows all mail to `user@domain.com` to pass through the conversion channel and for the conversion channel to emit mail to that address without causing a mail loop. (Note: This is not a realistic example, as actual routing through the conversion channel is normally instead handling via the [CONVERSIONS mapping table](#) rather than through rewrite rules; but this sort

of rewrite rule approach could be used with other channels, such as a `tcp_scanner` type of channel.)

A summary of template substitutions and control sequences is presented in [Summary of template substitutions and control sequences](#).

Table 34.3 Summary of template substitutions and control sequences

Substitution sequence	Substitutes
\$D	Portion of domain specification that matched
\$G	Insert (first) <code>defaulthost</code> argument
\$nG	Insert nth element, counting from left to right, of <code>defaulthost</code> argument
\$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	Username from original address
\$0U	Local part (username) from original address, minus any subaddress
\$1U	Subaddress, if any, from local part (username) of original address
\$\$	Inserts a dollar sign (literal)
\$\$%	Inserts a percent sign (literal)
\$\$@	Inserts an at sign (literal)
\$\$\	Force substituted material to lowercase
\$\$^	Force substituted material to uppercase
\$_	Use original case (and do not do LDAP URL character encoding)
\$\$=	Force subsequent material to be properly quoted (encoded) according to LDAP URL syntax rules
\$.text.	Specify text to use as the rewrite result in cases of temporary LDAP lookup failures
\$\$..	Turn off special handling of temporary LDAP lookup failures: even temporary LDAP lookup failures are interpreted as rewrite rule failure
\$\$W	Substitutes in a random, unique string
\$\$<...>	Substitute in a hash of the argument
\$\$n<...>	Substitute in a hash of the argument, modulo <i>n</i>
\$\$]...[
\$(text)	General "database" substitution; rule fails if lookup fails
\$\${...}	Apply specified mapping to supplied string
\$\$[...]	Invoke customer supplied routine; substitute in result
\$\$&n	<i>n</i> th part of unmatched (or wildcarded) host as counting from left to right starting from 0
\$\$!n	<i>n</i> th part of unmatched (wildcarded) host as counted from right to left starting from 0
\$\$*n	<i>n</i> th part of matching pattern as counting from left to right starting from 0
\$\$#n	<i>n</i> th part of matching pattern as counted from right to left starting from 0

Rewrite rule template substitutions
and control sequences

<code>\$nD</code>	Portion of domain specification that matched, preserving from the <i>n</i> th leftmost part starting from 0
<code>\$nH</code>	Portion of host/domain specification that didn't match, preserving from the <i>n</i> th leftmost part starting from 0
<code>\$Y</code>	Equivalent to <code>\$1Y</code> ; that is, substitute in the 1st (counting from the left, starting from 0) field of the transport information; for the case of incoming SMTP messages, this corresponds to the SMTP server IP address
<code>\$nY</code>	Substitute in the <i>n</i> th (counting from the left, starting from 0) field of the transport information; hence, for incoming SMTP messages <code>\$0Y</code> substitutes in the literal string "TCP", <code>\$1Y</code> (see also <code>\$Y</code>) substitutes in the SMTP server IP address, <code>\$2Y</code> substitutes in the SMTP server port, <code>\$3Y</code> substitutes in the SMTP client IP address, and <code>\$4Y</code> substitutes in the SMTP client port
Control sequence	Effect on rewrite rule
<code>\$1~</code>	Force channel match "success" effect, although truncating the rewrite rule at this point if the "real" channel match failed
<code>\$,</code>	(New in MS 7.0.5) Rewrite rule succeeds only if the top-level part of the domain name argument following this metacharacter is present in the current list of known Top Level Domains (the list as constructed from the <code>tlsds.txt</code> file)
<code>\$></code>	(New in MS 7.0.5) Rewrite rule succeeds only if the top-level part of the domain name argument following this metacharacter is not present in the current list of known Top Level Domains (the list as constructed from the <code>tlsds.txt</code> file)
<code>\$:</code>	Apply only if the address being rewritten is the result of an alias
<code>;\$</code>	Fail if the address being rewritten is the result of an alias
<code>\$E</code>	Apply only to envelope addresses
<code>\$B</code>	Apply only to header/body addresses
<code>\$F</code>	Apply only to forward-directed (<i>e.g.</i> , To:) addresses
<code>\$R</code>	Apply only to backwards-directed (<i>e.g.</i> , From:) addresses
<code>\$Mchannel</code>	Apply only if channel <i>channel</i> is rewriting the address
<code>\$1M</code>	Apply only if an "internal" channel is rewriting the address
<code>\$Nchannel</code>	Fail if channel <i>channel</i> is rewriting the address
<code>\$1N</code>	Fail if an "internal" channel is rewriting the address
<code>\$Qchannel</code>	Apply if sending to channel <i>channel</i>
<code>\$Cchannel</code>	Fail if sending to channel <i>channel</i>
<code>\$S</code>	Apply if host is from a source route
<code>\$A</code>	Apply if host is to the right of the at sign
<code>\$P</code>	Apply if host is to the right of a percent sign
<code>\$X</code>	Apply if host is to the left of an exclamation point
<code>\$Tnewtag</code>	Set the rewrite rule tag to <i>newtag</i>
<code>\$Ilist-name</code>	Apply only if the <i>list-name</i> matches
<code>\$Olist-name</code>	Fail if the <i>list-name</i> matches
<code> \$?errmsg</code>	If rewriting fails, return <i>errmsg</i> instead of the default error message

<code>\$nxxxyyy?errmsg</code>	If rewriting fails, return SMTP error code <i>n.xxx.yyy</i> and error text <i>errmsg</i> instead of the default SMTP error code and error message
<code>\$Vdomain</code>	Succeed if domainMap succeeds (that is, if the domain name <i>domain</i> is found in the LDAP directory)
<code>\$Zdomain</code>	Succeed if domainMap fails (that is, if the domain name <i>domain</i> is <i>not</i> found in the LDAP directory)
<code>\$nJ</code>	Set mailbox flags
<code>\$nK</code>	Clear mailbox flags
<code>\$nT</code>	Set override <code>alias_magic</code> value to be used for the domain which matched this rewrite rule when the rewrite rule is being used during alias expansion; <i>n</i> must be an appropriate <code>alias_magic</code> value
<code>\$n=B</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is the message block size
<code>\$n>B</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than the message block size
<code>\$n>=B</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than or equal to the message block size
<code>\$n<B</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than the message block size
<code>\$n<=B</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than or equal to the message block size
<code>\$n<>B</code>	(New in MS 8.0) Rewrite fails when <i>n</i> is the message block size
<code>\$n=L</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is the number of lines in the message
<code>\$n>L</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than the number of lines in the message
<code>\$n>=L</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than or equal to the number of lines in the message
<code>\$n<L</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than the number of lines in the message
<code>\$n<=L</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than or equal to the number of lines in the message
<code>\$n<>L</code>	(New in MS 8.0) Rewrite fails when <i>n</i> is the number of lines in the message
<code>\$n=P</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is the message priority
<code>\$n>P</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than the message priority
<code>\$n>=P</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is greater than or equal to the message priority
<code>\$n<P</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than the message priority
<code>\$n<=P</code>	(New in MS 8.0) Rewrite fails unless <i>n</i> is less than or equal to the message priority
<code>\$n<>P</code>	(New in MS 8.0) Rewrite fails when <i>n</i> is the message priority

34.4.2.1 Rewrite username and subaddress substitutions, \$U, \$0U, \$1U

Any occurrences of \$U in the [template](#) are replaced with the username (local part) from the original address. Note that usernames of the form a."b" will be replaced by "a.b" as current Internet standardization work is deprecating the former syntax from [RFC 822](#) and it is expected that the latter usage will become mandatory in future.

Any occurrences of \$0U in the template are replaced with the username from the original address, minus any [subaddress](#) (and [subaddress indication character](#) such as +). Any occurrences of \$1U in the template are replaced with the subaddress and subaddress indication character, if any, from the original address. (See the discussion of the [subaddresssexact](#) channel option and [Subaddresses in aliases](#) for background on subaddresses.) So note that \$0U and \$1U are complementary pieces of the username, with \$0U \$1U being equivalent to a simple \$U.

\$0U and \$1U might be used when it is desired to force the account portion of the local-part to lowercase, while retaining original case in the subaddress since the subaddress indicates a folder name. For instance, a rewrite rule:

```
org.domain.com    $\$0U$_$1U@org.domain.com
```

will cause an address such as nAmE@org.domain.com to be transformed (rewritten) to name@org.domain.com, while an address such as nAmE+sUbAdDrEsS@org.domain.com would be transformed to name+sUbAdDrEsS@org.domain.com.

34.4.2.2 Rewrite 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. \$nH and \$nD are variants that preserve the normal \$H or \$D portion from the *n*th leftmost part starting counting from 0. Or another way of putting it is that \$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, suppose the address jdoe@host.domain.com matches the rewrite rule

```
host.domain.com    $U%$1D@TCP-DAEMON
```

Then the result of the rewrite rule will be jdoe@domain.com with TCP-DAEMON used as the outgoing channel. Here where \$D would have substituted in the entire domain that matched, host.domain.com, the \$1D instead substitutes in the portions of the match starting from part 1 (part 1 being domain), so substitutes in domain.com.

\$L substitutes the portion of a domain literal that was not matched by the rewrite rule.

34.4.2.3 Rewrite subdomain single field substitutions, \$&n, \$!n, \$*n, \$#n

Subdomain single field substitutions extract a single subdomain part from the host/domain specification being rewritten. The available single field substitutions are shown in [Single field substitutions](#).

Table 34.4 Single field substitutions

Control Sequence	Usage
------------------	-------

<code>\$&n</code>	Substitute the n th element, $n=0,1,2,\dots,9$, in the host specification (the part that did not match explicit text in the pattern or matched a wildcard of some kind). Elements are separated by dots; the first element on the <i>left</i> is element zero. The rewrite fails if the requested element does not exist.
<code>#!n</code>	Substitute the n th element, $n=0,1,2,\dots,9$, in the host specification (the part that did not match explicit text in the pattern or matched a wildcard of some kind). Elements are separated by dots; the first element on the <i>right</i> is element zero. The rewrite fails if the requested element does not exist.
<code>*n</code>	Substitute the n th element, $n=0,1,2,\dots,9$, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the <i>left</i> is element zero. The rewrite fails if the requested element does not exist.
<code>#n</code>	Substitute the n th element, $n=0,1,2,\dots,9$, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the <i>right</i> is element zero. The rewrite fails if the requested element does not exist.

Suppose the address `jdoe@msgstore.domain.com` matches the rewrite rule

```
*.DOMAIN.COM      $U%$&0.domain.com@mailhub.domain.com
```

Then the result from the template will be `jdoe@msgstore.domain.com` with `mailhub.domain.com` used as the routing system.

34.4.2.4 Rewrite defaulthost substitutions, \$G, \$nG

Any occurrences of `$G` are replaced with the value of the first argument of the `defaulthost` channel option setting on the current source channel. Any occurrences of `$nG` are replaced with the value of the n th element, counting from zero, of the first argument of the `defaulthost` channel option; thus if `defaulthost a.b.c.d.com` is set on the current source channel, `$0G` will substitute "a", `$1G` will substitute "b", *etc.* If `defaulthost` is not set on a channel, then `$G` or `$nG` rewrite rules will fail (will not be considered to have matched).

34.4.2.5 Rewrite literal character substitutions, \$\$, \$%, \$@

The `$`, `%`, and `@` characters are normally metacharacters in rewrite rule templates. To insert a literal such character, quote it with a dollar character, `$`. *I.e.*, `$$` 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).

34.4.2.6 Rewrite case control substitutions, \$\\, \$^, \$ _

Character case in templates is normally preserved. In addition to preservation of the case of the literal text in a template, note that substitution sequences such as `$U` or `$D` that substitute material extracted from original addresses also preserve the original case of that material.

When it is desirable to force substituted material to use a particular case, for instance, to force mailboxes to lowercase on UNIX systems, special substitution sequences can be used in templates to force substituted material to a desired case. Specifically, `$\` forces subsequent substituted material into lower case, `$$` forces subsequent substituted material into upper case, and `$ _` says to use the original case (as well as turning off [LDAP URL character encoding](#)). So you can use a rule such as

```
unix.domain.com    $\$U$_%unix.domain.com
```

to force mailboxes to lowercase for unix.domain.com addresses.

34.4.2.7 Rewrite LDAP query URL substitutions, \$]...[, \$=, \$.text., \$..

A substitution of the form `$]ldap-url[` is handled specially. `ldap-url` is interpreted as an LDAP query URL and the result of the LDAP query is substituted. (If the LDAP query fails, it is as if the rewrite rule never matched in the first place.) Standard LDAP URLs as per [RFC 2255](#) are used, with the host and port typically omitted; the host and port are instead typically specified via the `ldap_host` and `ldap_port` MTA options. (Indeed, prior to MS 7.0u4 the host and port could not be specified in the URL itself; as of MS 7.0u4 explicitly specifying the host and/or port in the URL is supported.) That is, the LDAP URL should be specified as

```
ldap:///dn[?attributes[?scope?filter]]
```

or if specifying the LDAP host and LDAP port explicitly

```
ldap://ldap-host:ldap-port/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 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.

For instance, at a site `domain.com` with an LDAP server running on port 389 of the system `ldap.domain.com`, a legacy configuration MTA option file might set

```
LDAP_HOST=ldap.domain.com
LDAP_PORT=389
```

or in Unified Configuration

```
msconfig> show mta.ldap_host
role.mta.ldap_host = ldap.domain.com
msconfig> show mta.ldap_port
msconfig> show -default mta.ldap_port
mta.ldap_port: 389
```

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:

```
.domain.com \
    $U$H$D@$]ldap:///o=domain.com?mailRoutingSystem?sub?(mailDomain=$D)[
```

where here the rewrite substitution sequence \$D is used to substitute in the current domain name into the LDAP query constructed; for ease in reading, the backslash character, \, is used to continue the single logical rewrite rule line onto a second physical line.

Note that LDAP URLs have special character quoting (encoding) requirements. The \$= metacharacter forces subsequent material to be properly quoted (encoded) for LDAP URL usage as shown in [LDAP character encoding rules](#). Note that the "leave case as-is" substitution, \$_, discussed in [Rewrite case control substitutions](#) can be used to turn off LDAP URL character encoding.

Table 34.5 LDAP character encoding rules

Original character	Quoted (encoded) version
	%20
\$	%24
&	%26
(%5C28
)	%5C29
*	%5C2A
+	%2B
,	%2C
:	%3A
;	%3B
=	%3D
?	%3F
\	%5C5C

That is, any of the characters

\$ &+, : ; = ?

will be converted to the percent character, "%", followed by the hexadecimal representation of their location in US-ASCII; any of the characters

() *

will be converted to "%5C" followed by the hexadecimal representation of their location in US-ASCII (the encoded form of the backslash character followed by the hexadecimal for the particular character); while the backslash character itself

\

will be converted to "%5C5C".

The overall length of the LDAP URL (after any substitutions are performed) is limited to 252 characters in iMS 5.2, limited to 256 characters in JES MS 6.0 through JES MS 6.2, and limited to 1024 characters as of JES MS 6.3. Note also that the length of the original template

in which such an LDAP URL appears is limited: to 252 characters in iMS 5.2 and earlier, or to 256 characters as of JES MS 6.0 and later; but substitutions in the template, and in particular substitutions used to construct the LDAP URL, may increase the LDAP URL length.

By default, temporary LDAP failures cause the current rewrite rule to fail. This is problematic in cases where different actions need to be taken depending on whether the LDAP lookup failed to find anything versus the directory server being unavailable or misconfigured. As of MS 6.3, the `$.` metacharacter sequence can be used in a rewrite rule to establish a string which will be processed as the rewrite rule result in the event of a temporary LDAP lookup failure. The temporary failure string is terminated by an unescaped `"`. Thus the format is `$.text` in its complete form.

For rewrite rules, (unlike the similar `$.text` mechanism available in [mapping tables](#) where the temporary failure string is "sticky"), a temporary failure string remains set only for the duration of the current rewrite rule. `$.` can be used to return to the default state where no temporary failure string is set and temporary LDAP failures cause 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.

34.4.2.8 Rewrite general database substitutions, `$(...)`

A substitution of the form `$(text)` is handled specially. The `text` part is used as a key to access the MTA's [general database](#). If `text` is found in the database, then the corresponding template from the database is substituted. If `text` does not match an entry in the database, then the rewrite process fails; it is as if the rewrite rule never matched in the first place. If the substitution is successful, then the template extracted from the database is re-scanned for additional substitutions. However, additional `$(text)` substitutions from the extracted template are prohibited in order to prevent endless recursive references.

Depending upon the setting of the MTA option `use_text_databases`, the general "database" is either stored and accessed as an on-disk database (formerly the default; now deprecated), or as an in-memory structure constructed (during configuration compilation or MTA initialization) from an on-disk flat text file. The on-disk database, if that is what is being used, is `IMTA_ROOT:data/db/generalldb` (which formerly could be redirect via the now-deleted `imta_general_database` MTA Tailor option), which must be generated using the `imsimta crdb` utility from some site-supplied source text file. If an in-memory database structure is instead being used, then when the MTA configuration is compiled (or at MTA process initialization time, if a compiled configuration is not in use) the MTA reads the `IMTA_ROOT:config/general.txt` file (which formerly could be redirected via the now-deleted `imta_general_data` MTA Tailor file option) and compiles it into an in-memory structure. Use of an in-memory "database" is normally recommended (for reasons of performance and reliability); however, do note that use of this in-memory "database" does require recompiling the configuration to get changes to the "database" (changes to the source text file) incorporated into the compiled configuration.

As an example, suppose that the address `jdoe@host1.domain.privateuse` matches the rewrite rule

```
.privateuse      $( $H )
```

Then, the text string `host1.domain` will be looked up in the general database and the result of that look up, if any, instead used for the rewrite rule's template. Suppose that the result

of looking up `host1.domain` is `$u%mailhost.domain.com@tcp_intranet-daemon`. Then the output of the template will be `jdoe@mailhost.domain.com` (*i.e.*, `username = jdoe`, `host/domain specification = mailhost.domain.com`), and the routing system will be `tcp_intranet-daemon` (the [typical official host name of the `tcp_intranet` channel](#)).

If a general database exists it should be world readable to insure that it operates properly.

34.4.2.9 Rewrite apply specified mapping substitutions, `{ . . . }`

A substitution of the form `{ mapping , argument }` is handled specially. The `mapping , argument` part is used to find and apply an [MTA mapping table](#). 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`, then the rewrite will fail. If successful, the result of the mapping is merged into the template at the current location and reexpanded.

This mechanism allows the MTA's 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's rewriting process is capable of.

34.4.2.10 Rewrite routine substitutions, `[. . .]`

A substitution of the form `[image , routine , argument]` is handled specially. The `image , routine , argument` part is used to find and call an Oracle-supplied or customer-supplied routine. At run-time 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)
```

where **argument** and **result** are 256 byte long (252 byte long in iMS 5.2 and earlier) character string buffers. On Solaris, **argument** and **result** are passed as a pointer to a character string, (*e.g.*, 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 `routine` should return 0 if the rewrite rule should fail and -1 if the rewrite rule should succeed.

This mechanism allows the MTA's 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. For instance, directory service lookups for forward pointing addresses (*e.g.*, `To: addresses`) to the host `domain.com` might be performed as follows with the following rewrite rule (the `$F`, described in [Address direction and location-specific rewrites](#) causes this rule to only be used for forward pointing addresses):

```
domain.com      $F$[LOOKUP_IMAGE,LOOKUP,$U]
```

A forward pointing address `jdoe@domain.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 `"jdoe"` as the **argument** parameter. The routine `LOOKUP` might then return a different address, say, `John.Doe%specialhost.domain.com` in the **result** parameter and the value -1 to indicate that the rewrite rule succeeded. The percent sign in the result

string causes, as described in [Repeated rewriting template](#), the rewriting process to start over again using John.Doe@specialhost.domain.com as the address to be rewritten.

On UNIX systems, the site-supplied shared library image `image` should be world readable.

Note: This facility is not designed for use by casual users; it is intended to be used to extend the MTA's capabilities system-wide.

34.4.2.11 Rewrite unique string substitution, `$W`

Each use of the `$W` substitution inserts a text string composed of upper case letters and numbers that is designed to be unique and unrepeatable. `$W` is useful in situations where nonrepeating address information must be constructed.

34.4.2.12 Rewrite hash substitutions, `$<...>`, `$n<...>`

`$<text>` substitutes a hash of `text`. `$n<text>` substitutes a hash of `text`, modulo `n`.

34.4.2.13 Rewrite transport substitutions, `$Y`, `$nY`

`$nY` substitutes in the `n`th field of the transport information, with a plain `$Y` being synonymous with `$1Y`. Transport information is as it would appear in [PORT_ACCESS mapping table probes](#), or optionally various other access mapping tables as enabled via the [include_connectioninfo](#) MTA option, or be logged in MTA transaction log entries, as controlled by the [log_connection](#) MTA option (bit 1/value 2 causes generation of connection transaction entries, which normally include transport information, and bit 3/value 8 caused inclusion of transport information in message transaction log entries); that is, transport information has the format:

```
TCP | server-address | server-port | client-address | client-port
```

Hence for incoming SMTP/SMTP SUBMIT messages, `$0Y` substitutes in the literal string "TCP"; `$1Y` (or `$Y` which is synonymous) substitutions in the SMTP/SMTP SUBMIT server IP address; `$2Y` substitutes in the SMTP/SMTP SUBMIT server port; `$3Y` substitutes in the SMTP/SMTP SUBMIT client IP address; and `$4Y` substitutes in the SMTP/SMTP SUBMIT client port.

If `$Y` and `$nY` is used in a rewrite rule template but transport information is not available for the address being rewritten, the rewrite rule will fail.

34.4.2.14 Source channel-specific rewrites, `$M`, `$N`, `$1M`, `$1N`

It is possible to have rewrite rules that act only in conjunction with specific source channels. This can be useful when a hostname has a different meaning when it appears in a message arriving on one channel and another when it appears in a message arriving on a different channel -- such name "collisions" used to arise in Bitnet days. Source channel-specific rewrite rules can also be useful when it is desired to achieve different routing for messages coming in on different channels:^a perhaps routing through a spam/virus scanner box for messages from some "untrusted" source.

Source channel-specific rewriting is associated with the channel that is operating and the channel options [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, channel-specific rule checks are enforced. `rules` is the default.

Source channel-specific rewriting is *not* associated with the channel a given address matches. It depends only on the MTA component doing the rewriting and that component's channel table entry.

Source channel-specific rewrite checking is triggered by the presence of a \$N or \$M control sequence in the [template](#) part of a rewrite 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.

`$Mchannel` causes the rule to fail if the channel *channel* is not currently doing the rewriting. `$Nchannel` causes the rule to fail if the channel *channel* is doing the rewriting.

Multiple \$M and \$N clauses may be specified. If any one of multiple \$M clauses matches, the rule will succeed. If any of multiple \$N clauses matches, the rule will fail.

The `$1M` control sequence causes the rule to fail if a non-"internal" channel is currently doing the rewriting; that is, the rule will succeed only if an "internal" channel is doing the rewriting. The `$1N` control sequence causes the rule to fail if an "internal" channel is currently doing the rewriting; that is, the rule will succeed only if a non-"internal" channel is doing the rewriting. ("Internal" here means internal-to-the-MTA: a [reprocess](#), [process](#), or [conversion channel](#) type of channel; it does *not* mean a `tcp_intranet` type of channel.)

^a Alternatively, and often more conveniently, source-specific routing can be achieved via the [CONVERSIONS mapping table](#).

34.4.2.15 Destination channel-specific rewrites, \$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 a message is being enqueued and the channel options [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. `rules` is the default.

Destination channel-specific rewriting is *not* associated with the channel a given address matches. 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 will be enqueued. *During the rewriting of the envelope To address any \$C and \$Q control sequences are ignored.* Once 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.

`$Qchannel` causes the rule to fail if the channel *channel* is not the destination. `$Cchannel` causes the rule to fail if the channel *channel* is the destination.

Multiple \$Q and \$C clauses may be specified. If any one of multiple \$Q clauses matches, the rule will succeed. If any of multiple \$C clauses matches, the rule will fail.

For example, suppose the local host's [TCP/IP channel used to communicate with the Internet is the tcp_local channel](#). Then, to prevent "raw" user@host.bitnet style addresses from appearing on messages queued to that channel, a rewrite rule of the form

```
.BITNET      $U$%$H$D@interbit.cren.net$Qtcp_local
```

might be used. This will, in messages destined to the tcp_local channel, transform addresses of the form user@host.bitnet to user%host.bitnet@interbit.cren.net.

34.4.2.16 Address direction and location-specific rewrites, \$B, \$E, \$F, \$R

It is sometimes useful to specify rewrite rules that only apply to envelope addresses or, alternately, only apply 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 an envelope To address or an address that originates on a To:, Cc:, Resent-to:, or other header line that refers to a destination. A backwards-pointing address is an envelope From address or an address from a header line such as From:, Sender:, or Resent-From:, referring to a source. The control sequence \$F causes the rewrite to fail if the address is backwards-pointing. The control sequence \$R causes the rewrite to fail if the address is forward-pointing.

The first of the following rewrite rules causes forward pointing envelope addresses (*i.e.*, envelope To addresses) of the form user@oldhost.domain.com to be rewritten to user@newhost.domain.com and the message routed to the tcp_intranet channel:

```
oldhost.domain.com    $U%newhost.domain.com@tcp_intranet-daemon$E$F  
oldhost.domain.com    $U@domain.com
```

All other, non-envelope-To occurrences, of addresses of the form user@oldhost.domain.com are rewritten to user@domain.com by the second rewrite rule.

34.4.2.17 Host location-specific rewrites, \$A, \$P, \$S, \$X

Circumstances occasionally require rewriting that's 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, or 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. Situations can arise, however, which may necessitate 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 may match a host extracted from a source route, \$A specifies that the rule may match a host found to the right of the at sign, \$P specifies that the rule may match a host found to the right of a percent sign, and \$X specifies that the rule may match a

host found to the left of an exclamation point. The rule will fail if the host is from a location other than one specified.

These sequences can be combined in a single rewrite rule. For example, if \$S and \$A are specified the rule will match 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.

34.4.2.18 Domain LDAP lookup rewrites, \$V, \$Z

The \$V and \$Z flags interpret the material following (up to the first @ or % character, or \$C, \$M, \$N, \$Q, or \$T) as a domain name to look up in the LDAP directory; (in Schema 1, this would be a lookup in the DC tree within the directory; in Schema 2, domains are stored as part of the Organization tree so it is a lookup in the Organization tree). \$V means succeed if the LDAP lookup of the domain succeeds (*i.e.*, the domain is found, as a local/hosted/vanity domain). \$Z means succeed if the LDAP lookup of the domain fails (*i.e.*, the domain is not a local/hosted/vanity domain).

For instance, a typical Oracle Messaging Server MTA configuration will include a rewrite rule:

```
$*      $A$E$F$U%$H$V$H@local-channel-official-hostname
```

where *local-channel-official-hostname* corresponds to the value of `channel:1.official_host_name`. Note that this fundamental rewrite rule of [Direct LDAP configuration](#) makes use of the [Initial match-all rule, \\$*](#), so that it is the very first rewrite rule checked for any domain name appearing to the right of the @ sign ([\\$A control sequence](#)) in an envelope To address ([\\$E and \\$F control sequences](#)).

Note that in Unified Configuration, this same rewrite rule would typically be expressed using the [&/IMTA_HOST/ substitution](#), so appear as:

```
msconfig> show rewrite * "$*"
role.rewrite.rule = $* $A$E$F$U%$H$V$H&/IMTA_HOST/
```

The LDAP server to query, as well as other basic LDAP query parameters relevant in domainMap lookups, are controlled by certain MTA options and/or (in legacy configuration) `configutil` parameters; see [Basic configuration settings relevant to domain LDAP lookups](#). The MTA options, if explicitly set, take precedence over (override) their corresponding `configutil` parameters.

Table 34.6 Basic configuration settings relevant to domain LDAP lookups

msconfig base option	configutil parameter	MTA option	Default	Description
ugldaphost	<code>local.ugldaphost</code>	ldap_host	++	LDAP host to which to connect
ugldapport	<code>local.ugldapport</code>	ldap_port	389	LDAP port to which to connect
ldapsearchtimeout+	<code>local.ldapsearchtimeout+</code>	ldap_timeout	180000	Time out, in seconds, for LDAP queries
ugldapbinddn	<code>local.ugldapbinddn</code>	ldap_username		The username with which to bind when doing LDAP queries
ugldapbindcred	<code>local.ugldapbindcred</code>	ldap_password		The password with which to bind when doing LDAP queries
ldaprequiretls			0	(New in 8.0) If SSL is not already being used on a given LDAP connection (<i>e.g.</i> , due to ugldapusessl being set or use of an ldaps: URL), then enabling <code>base.ldaprequiretls</code> will require successful negotiation of TLS (using

Rewrite rule template substitutions and control sequences

				LDAP StartTLS) before proceeding with the connection.
		<code>ldap_max_connections</code>	1024	The maximum number of simultaneous LDAP connections to allow using
<code>defaultdomain</code>	<code>service.defaultdomain</code>	<code>ldap_default_domain</code>		The default domain name
<code>dcroot</code>	<code>service.dcroot</code>	<code>ldap_domain_root</code>	<code>o=internet</code>	The base DN for the domain portion of the DIT
	<code>local.imta.schematag</code>	<code>ldap_schematag</code>	<code>ims50</code>	The tag for the schema in use
<code>ldap_domain_filter_schema1</code>		<code>ldap_domain_filter_schema1</code>	<code>((objectclass=inetDomain)(objectclass=inetdomainalias))</code>	Specifies the filter for domains when schema 1 is in use
<code>ldap_domain_filter_schema2</code>		<code>ldap_domain_filter_schema2</code>		Specifies the filter for domains when schema 2 is in use
<code>ldap_domain_known_attributes</code>		<code>ldap_domain_known_attributes</code>	-1	This option controls whether the MTA requests the return of all domain attributes, or (the default) requests the return of only "known" domain attributes, specifically the per-domain attributes listed in Table of MTA LDAP attribute name options
		<code>domain_match_url</code>		Specify an additional LDAP query URL to attempt if a domain name cannot be found as a "real" domain; for instance, this option would be set to <code>ldap:///SB?msgVanityDomain?sub?(msgVanityDomain=\$D)</code> if one wishes to support vanity domains
		<code>domain_uplevel</code>	0	This option affects how domain names are searched for and used; in particular, it controls whether the MTA iteratively looks "up" for a domain when a subdomain cannot be found
		<code>domain_failure</code>	<code>reprocess-daemon \$Mtcp_local\$1M \$1--error\$4000000? Temporary lookup failure</code>	What rewrite template to use if a \$V or \$Z rewrite rule lookup encounters an LDAP error, such as an LDAP connection error
<code>ldap_domain_timeout</code>		<code>ldap_domain_timeout</code>	900	Time (in seconds) to retain cached results of domain lookups (in the domain map library code cache)
		<code>domain_match_cache_size</code>	100000	Number of domain lookup results to cache (in the MTA's cache)
		<code>domain_match_cache_timeout</code>	600	Time (in seconds) to retain (in the MTA's cache) cached results of domain lookups

+The `ldapsearchtimeout` base option (Unified Configuration) or `local.ldapsearchtimeout` `configutil` parameter (legacy configuration) is a global default for all searches done through the LDAP pool API, including those done by the MTA.

++The MTA option `ldap_host` defaults to the value of the `ugldaphost` base option, which in turn defaults, if not set, to the loopback interface.

Compare this [Basic configuration settings relevant to domain LDAP lookups](#) with [Table of Basic configuration settings relevant to alias LDAP lookups](#).

The `domain_match_url` and `domain_uplevel` MTA options further affect domain lookups, with `domain_match_url` potentially specifying an additional lookup to look for vanity domains (which are not real domains), and with `domain_uplevel` controlling things such as whether if a subdomain is not found, the MTA then looks instead for the domain "over" the subdomain.

If a \$V or \$Z lookup attempt encounters an LDAP error condition (such as the LDAP directory being temporarily inaccessible), then the MTA option `domain_failure` specifies what the MTA will take to be the rewriting process result. The default value for `domain_failure` means that LDAP error conditions will result in messages being diverted to the [reprocess channel](#) for additional subsequent rewriting and lookup attempts.

The results of a domain name lookup due to a \$V and \$Z flag will be cached; that is, the MTA caches not only whether the domain name lookup was successful, but also (in the

case of a successful lookup) any attribute values successfully returned. In its queries, the MTA can request that successful lookups return either all attributes for the domain, or instead request an explicit list of "known to the MTA attributes" (see the per-domain attributes in [Table of MTA LDAP attribute name options](#)); note that for some directory setups, there may be an LDAP directory performance difference between requesting all attributes or requesting an explicit (even large explicit) list of attributes. Whether domain name lookup requests are for all attributes, or a list of known attributes, is controlled by the [ldap_domain_known_attributes](#) MTA option; the default is to request the return of all domain attributes. For control of domain name lookup result caching at the MTA-level, see the [domain_match_cache_size](#) and [domain_match_cache_timeout](#) MTA options; note that the underlying domain Map code also does its own caching, with timeout (when called by the MTA) controlled by the [ldap_domain_timeout](#) MTA option.

34.4.2.19 Channel match force truncation rewrite, \$1~

The special-purpose \$1~ rewrite rule control sequence is normally used in conjunction with the [domain_failure](#) MTA option.

The \$1~ control sequence modifies the effect of channel matching checks, such as the [\\$C](#), [\\$Q](#), [\\$M](#), and [\\$N](#) channel match checks. Normally such checks either succeed, in which case the rewrite rule is used, or fail, in which case it is as if the rewrite rule never even matched and the rewrite rule is not used at all for the address in question. The \$1~ control sequence is used when it is desired to instead have the rewrite rule always be used, but with a different right hand side depending upon whether the initial channel match succeeded or failed. That is, the \$1~ control sequence overrides the original channel match results (forces a channel match success state), and then either truncates the rewrite rule template (if the original channel match check failed), or allows additional material to be suffixed to the rewrite rule template (if the original channel match check succeeded).

For instance, given channel definitions such as

```
tcp_local smtp mx ...rest-of-keywords...
tcp-daemon
```

```
tcp_domainrelay smtp mx ...rest-of-keywords...
tcp-daemon-domainrelay
```

then a rewrite rule such as

```
domain.com      $U%domain.com@tcp-daemon$Mtcp_local$1~-domainrelay
```

means that when the `tcp_local` channel is rewriting a `domain.com` address, then the effective template used will be

```
$U%domain.com@tcp-daemon-domainrelay
```

whereas when any other channel is rewriting a `domain.com` address, then the effective template used will be

```
$U%domain.com@tcp-daemon
```

That is, messages coming in the `tcp_local` channel addressed to `domain.com` will be routed out the `tcp_domainrelay` channel, whereas messages coming in from any non-`tcp_local` channel addressed to `domain.com` will be routed out the `tcp_local` channel.

The `!~` control sequence is typically used in the value of the `domain_failure` MTA option; see the discussion of that option which includes another example of use of `!~`.

34.4.2.20 TLD comparison rewrites, `!`, and `!>`

New in 7.0.5, the `!`, and `!>` control sequences allow checking domain top level portions against the current¹ TLD (Top Level Domain) list. They act like `!V` and `!W`, respectively, except that instead of checking for a domain known to the directory, they check the top-level part of the current domain against the list of known valid TLDs. `!` succeeds if the top-level part is on the list; `!>` succeeds if the top-level part isn't on the list.

¹To keep the `tllds.txt` file on your host up-to-date, it is recommended that you regularly fetch a new version. See, for instance, <https://data.iana.org/TLD/tlds-alpha-by-domain.txt>. After fetching a new `tllds.txt` file, issue the command `imsimta chbuild` to have the MTA rebuild its list.

34.4.2.21 `alias_magic` override rewrite, `!nT`

Rewrite rules can override the `alias_magic` MTA option setting and `aliasmagic` (source) channel option setting. Specifically, a construct of the form `!nT`, 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.

34.4.2.22 Alias-sensitive rewrites, `!:` and `!;`

It is possible to have rewrite rules whose application is dependent upon whether or not the address being rewritten was the result of an `alias`. (Note that inherently, only envelope To addresses can possibly be the result of an alias.) This can be useful in the case, for instance, where certain rewrite rule(s) should be applied only before alias expansion has occurred, while other rewrite rule(s) should be applied only after alias expansion has occurred.

Alias-sensitive rewrite checking is triggered by the presence of the `!:` or `!;` control sequence in the template part of a rule. The `!:` control sequence means that the rewrite rule will match only if the address being rewritten is the result of an alias. (So note that such a rewrite rule also is implicitly a `!E!F` rewrite rule---it can only ever match envelope To addresses.) The `!;` control sequence means that the rewrite rule will match only if the address being rewritten is not the result of an alias.

34.4.2.23 List-name-sensitive rewrites, `!I`, `!O`

Mailing lists defined in the MTA can have an associated list name, *list-name*, established via the `mgrpListName` LDAP attribute (or whatever LDAP attribute is named by the `ldap_list_name` MTA option), or via the `alias_list_name` alias option, or via the `[LIST_NAME] alias file named parameter`.

List name sensitive rewrite checking is triggered, when rewriting addresses that have a list name set, by the presence of a `!I` or `!O` control sequence in the template part of the rule. The characters following the `!I` or `!O`, up until either an at sign, percent sign, or subsequent `!N`, `!M`, `!C`, `!Q`, `!T`, or `!?`, are interpreted as a list name. If the address being rewritten has an

associated list name, then the list name specified must match that in a `$I` rewrite rule, or must not match that in a `$O` rewrite rule.

Note that if the address being rewritten has no associated list name, which would normally be the case for most addresses, then any list name check clauses in a rewrite rule are ignored.

34.4.2.24 Message size or priority comparison rewrites

As of MS 8.0, rewrite rules can be made sensitive to the expected size and/or priority (SMTP MT-PRIORITY; see [RFC 6710 \(SMTP Extension for Message Transfer Priorities\)](#)) of an incoming message. Such checking is triggered by the presence of a

`$NcomparisonC`

control sequence in the template part of a rule, where *N* is an appropriate numeric value (which note should be in the range -9 through 9 for MT-PRIORITY checks), *comparison* is one of =, >, >=, <, <=, or <>, and where *C* (the characteristic code) is one of B, L, or P (indicating message block size, message number of lines, and message priority, respectively).

So for instance the control sequences:

```
$100<B
$2000>=L
$4=P
```

would limit a rewrite rule to applying only when a message matched the conditions, respectively, (1) block size less than 100, (2) line size greater than or equal to 2000, (3) MT-PRIORITY value of 4.

34.4.2.25 Changing the current tag value, `$T`

The `$T` control sequence is used to change the current rewrite rule tag. The rewrite rule tag is prepended to all rewrite rule patterns before they are looked up in the configuration file/[rewrite group](#) and domain database. Text following the `$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          $$U@localhost$Ttcp-force |
tcp-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.

34.4.2.26 Controlling error messages associated with rewriting, \$?, \$nxxxyyy?

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 "unknown host or domain" (see the [error_text_unknown_host](#) MTA option). A special control sequence can be used to change the error message that will be printed if the rule fails. The sequence \$? is used to specify an error message. Text following the \$?, up until 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 will last 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 instance, if the final rewrite rule in the MTA configuration file is

```
.    $?Unrecognized address; contact postmaster@xyzyzy.com
```

then any unrecognized host/domain specifications which will fail will, in the process of failing, generate the error message "Unrecognized address; contact postmaster@xyzyzy.com".

Optionally, an extended SMTP error code may be included in the \$? template, controlling among other things whether the error returned is a temporary error, or a permanent error; the format is:

```
$nxxxyyy?error-text
```

where the n is either 4 (meaning a temporary error) or 5 (meaning a permanent error) and the xxx and yyy specify the second and third digits, respectively, of the extended SMTP error code. *E.g.*,

```
offline.domain.com    $4002001?Mailboxes$ temporarily$ unavailable;$ try$ later
```

will result in extended SMTP error code and error text "4.2.1 Mailboxes temporarily unavailable; try later". If an extended SMTP error code is specified but no *error-text* is specified, then the text of [error_text_temporary_failure](#) or [error_text_permanent_failure](#) will be used, as appropriate.

34.5 Domain database

The MTA's domain database, seldom used nowadays, provided an alternate location for storing [rewrite rules](#), for cases of exceptionally large numbers of rewrite rules.

In early versions of the MTA, the format of the domain database was an on-disk database, built using the [imsimta crdb utility](#) based upon a flat text file input. The allowed format of the flat text input file is:

key value

one entry per line, with the key beginning in column one, one or more white space (SP or TAB) characters, and then the value on the right hand side.

The `comment_chars` MTA option controls which characters (by default exclamation point and semicolon) in column one of a line are considered to indicate a comment line. The left angle character may be used to read another file into the domain database text input file.

For a `crdb` "on disk" database, the left hand side (the key) cannot contain spaces or tabs unless the `-quoted` switch is used; the maximum length of the key and value depend upon whether the `-long_records` switch is used.

New in the 8.0 release, the MTA supports use of memcache for certain database/storage uses, including the domain database; see the `domain_database_url` MTA option.

The `use_domain_database` MTA option controls whether or not the MTA makes use of the domain database. In MS 7.2 and earlier, the default for `use_domain_database` was 1, so the mere presence of the domain database file was enough to activate this database facility in the MTA. As of MS 7.3, the default for `use_domain_database` changed to 0 from the prior default of 1, so as of MS 7.3 it is necessary to explicitly set `use_domain_database=1` (and then if using a compiled configuration, recompile the configuration), and restart any long-running processes (*e.g.*, SMTP server processes) to enable consultation of the domain database. Note that the domain database is consulted only when no match was found among the rules in the `rewrite group` (Unified Configuration) or the configuration file (legacy configuration). That is, the domain database is only consulted if a given rule is *not* found in the `rewrite group` or configuration file, so rules can always be added to override those in the database.



Chapter 35 Aliases

35.1 Overview of Direct LDAP configuration	35-3
35.2 Aliases in LDAP	35-5
35.3 Aliases in Unified Configuration	35-8
35.4 Alias options	35-9
35.4.1 alias_entry alias option	35-9
35.4.2 alias_alterate_recipient Option	35-10
35.4.3 alias_and and alias_or alias options	35-10
35.4.4 alias_auth_channel and alias_cant_channel alias options	35-10
35.4.5 alias_*_list alias options	35-10
35.4.6 alias_auth_mapping and alias_cant_mapping alias options	35-10
35.4.7 alias_auth_username and alias_cant_username alias options	35-10
35.4.8 alias_autosecretary alias option	35-11
35.4.9 alias_blocklimit and alias_linelimit alias options	35-11
35.4.10 alias_capture and alias_journal alias options	35-11
35.4.11 alias_conversion_tag alias option	35-11
35.4.12 alias_creation_date alias option	35-12
35.4.13 alias_deferred* alias options	35-12
35.4.14 alias_*delay_notifications alias options	35-13
35.4.15 alias_digest_recurrence alias option	35-13
35.4.16 alias_direct_list and alias_direct_mapping alias options	35-14
35.4.17 alias_envelope_from alias option	35-14
35.4.18 alias_error_text alias option	35-14
35.4.19 alias_expandable and alias_nonexpandable alias options	35-15
35.4.20 alias_expiry alias option	35-15
35.4.21 alias_filter alias option	35-15
35.4.22 alias_header_addition and alias_header_trim alias options ..	35-15
35.4.23 alias_header_* alias expansion options	35-16
35.4.24 alias_header_check alias option	35-16
35.4.25 alias_hold_list, alias_nohold_list, alias_hold_mapping, alias_nohold_mapping alias options	35-16
35.4.26 alias_importance, alias_precedence, alias_priority, and alias_sensitivity alias options	35-17
35.4.27 alias_keep_delivery and alias_keep_read alias options	35-17
35.4.28 alias_list_name alias option	35-17
35.4.29 alias_moderator_* and alias_username_moderator_list alias options	35-17
35.4.30 alias_*originator_reply alias options	35-18
35.4.31 alias_received*, alias_noreceived* alias options	35-19
35.4.32 alias_nosolicit alias option	35-19
35.4.33 alias_optin alias option	35-19
35.4.34 alias_optinN alias options	35-19
35.4.35 alias_password alias option	35-20
35.4.36 alias_*_text alias options	35-20
35.4.37 alias_presence alias option	35-20
35.4.38 alias_private and alias_public alias options	35-20
35.4.39 alias_reprocess alias option	35-21
35.4.40 alias_sasl_* alias options	35-21
35.4.41 alias_sequence_* alias options	35-22
35.4.42 alias_single alias option	35-22
35.4.43 alias_spare* alias options	35-22

35.4.44	<code>alias_tag</code> alias option	35-23
35.4.45	<code>alias_toalias</code> option	35-23
35.4.46	<code>alias_username</code> alias option	35-23
35.5	Alias file	35-23
35.5.1	Alias file format	35-24
35.6	Alias database	35-42
35.6.1	Using another alias source and the alias database	35-42
35.6.2	Alias database format	35-43
35.7	Subaddresses in aliases	35-44
35.8	Alias special formats	35-45
35.9	Alias header addition modifiers	35-46
35.10	Alias recursion and nested list definitions	35-46
35.11	Alias restrictions	35-47
35.11.1	General alias restrictions	35-47
35.11.2	Additional LDAP alias restrictions	35-48
35.11.3	Additional alias file (or database) restrictions	35-48
35.12	Address reversal	35-48
35.12.1	LDAP lookups for address reversal	35-49
35.12.2	Reverse database	35-50
35.12.3	REVERSE mapping table	35-52
35.12.4	Subaddresses and address reversal	35-54
35.12.5	RFC 822 comment strings and personal name modification	35-55
35.13	Forwarding mail	35-57
35.13.1	Forwarding via user LDAP attributes	35-59
35.13.2	FORWARD mapping table	35-59
35.13.3	Forward database	35-61

As part of the MTA's central role of routing messages, the MTA must recognize and potentially transform addresses to determine whether to route them. Aliases -- addresses that translate to one or more other addresses -- are a fundamental part of the processing. The MTA supports various mechanisms for implementing aliases, and *many* variants on alias handling.

Aliases are used for "simple" forwarding (which may not be all that simple), for supporting "hosted domains", for defining "mail forwarders" (mail groups), for defining true mailing lists (which have additional semantics beyond mail groups), and for "centralized naming". The term *address reversal* is used for the process (and techniques) of esthetic transformations of addresses (as in the appearance of addresses in header lines) for purposes of centralized naming, or conforming to site addressing conventions.

Typical site provisioning nowadays is to store the bulk of user definitions (and therefore local domain definitions), as well as group and mailing list definitions, in an LDAP directory. The MTA supports such usage, often referred to as *Direct LDAP configuration*.

The MTA also supports the older style of defining user aliases via the MTA's *alias file* (legacy configuration), or *alias options* (Unified Configuration) -- techniques which are still used for special users (*e.g.*, postmaster) or special, limited purposes even in primarily LDAP-based configurations.

Note that the MTA limits aliases, and addresses in general, to a maximum of 256 characters. (Internet mailers were originally only required to support domain names up to a length of 63 characters, though support for lengths up to 255 characters was recommended, and similarly are only required to support a local-part -- that is, the part to the left of the "@" sign -- of up

to 64 characters. See for instance [RFC 1123, Requirements for Internet Hosts, Section 2.1](#), and [RFC 5321, SMTP, Section 4.5.3.1](#). Keep such limits in mind if you wish your addresses to work reliably on the Internet.) This limit of 256 characters is on the actual address itself; the [RFC 822 "phrase"](#) more commonly referred to as a personal name is a separate string with its own 256 character limit.

35.1 Overview of Direct LDAP configuration

A normal *Direct LDAP* configuration, as typically used at most sites nowadays, consists of provisioning local domain definitions and [local user entries in LDAP](#), plus optionally provisioning [mail group and mailing list entries](#) in LDAP, and then configuring the MTA to consult LDAP to find and use those domain and user (and group and mailing list) definitions.

Once domains and users (and optionally mail groups and mailing lists) are provisioned in LDAP, then configuring the MTA to make use of this information has five main steps:

1. *Inform the MTA where the LDAP directory resides.* In legacy configuration, the MTA consults the `local.ugldaphost` and `local.ugldapport` configutil parameters (which may be overridden specifically for the MTA's purposes by the MTA options `ldap_host` and `ldap_port`, respectively); in Unified Configuration, the MTA consults the base options `ugldaphost` and `ugldapport` (which similarly may be overridden for MTA purposes by the MTA options `ldap_host` and `ldap_port`). Various other configuration settings can further adjust aspects of the MTA's connections and consultations of LDAP, for instance, those discussed in [LDAP bind and connect MTA options](#).
2. *Configure the MTA to consult LDAP to determine which domains are "local", that is, which domains are hosted by this site.* This is achieved by a special rewrite rule, which in legacy configuration appears as:

```
$*    $A$E$F$U%$H$V$H@official-host-name-of-l-channel
```

or in Unified Configuration:

```
msconfig> show rewrite.rule * $*
role.rewrite.rule = $* $A$E$F$U%$H$V$H@&/IMTA_HOST/
```

This rewrite rule uses the match-all match-first `$*` template (so that it matches all domains and will be consulted before all other rewrite rules---see [Initial match-all rule](#)), but with `EF` in the template so that it applies only to envelope To addresses (see [Address direction and location-specific rewrites](#)), then uses a pattern that uses `VH` to look up the currently-being-rewritten-envelope-to domain in LDAP (see [Domain LDAP lookup rewrites](#)) to determine whether the domain is a "local" domain.

- a. If the domain is *not* "local" (is not found in LDAP), then the rewrite fails, and the envelope To address is routed per the rest of the rewrite rules.
- b. If the domain is "local" (is found in LDAP) but the domain is provisioned for override routing, as with the `mailRoutingHosts` LDAP attribute (or more precisely whatever LDAP attribute is named by the `ldap_domain_attr_routing_hosts` MTA option), then the envelope To address is converted to a form using source-routing to the routing host, and then routed per the rest of the rewrite rules (typically out a [tcp_intranet channel](#)).

- c. If the domain is "local" (found in LDAP) and has no override routing provisioned, then this rewrite rule forces this envelope To address to "match" the local channel. Such forcing of the recipient address to match-the-local-channel then sets the stage for step 3...
 3. *Configure the MTA to consult LDAP to find "local" recipient addresses.* Recipient (envelope To) addresses matching the local channel are looked up in LDAP via the [alias_url0](#) MTA option template to determine whether the recipient address corresponds to a valid "local" user (or group or mailing list).
 4. *Configure the interpretation of routing and delivery settings.* When a recipient (envelope To) address (user, group, or mailing list) is found in LDAP (step 3), then the MTA checks whether or not this MTA system should apply the recipient's `mailDeliveryOption` values, (more precisely, the values of the LDAP attribute named by the [ldap_delivery_option](#) MTA option), with interpretation of such `mailDeliveryOption` values performed as defined via the MTA's [delivery_options](#) option.
 - a. If the recipient address has a `mailHost` (MTA option [ldap_mailhost](#)) value set which does not match "this" host (as determined by comparing with the [ldap_local_host](#) and [ldap_host_alias_list](#) MTA options' values) and at least some of the recipient address's `mailDeliveryOption` clauses are "mailhost-sensitive", or if the source channel was configured with [aliasdetourhost](#) or [aliasoptindetourhost](#), then the address is converted to a source-routed form which explicitly routes to the `mailHost` or `aliasdetourhost` system.
 - b. If the recipient address has a `mailHost` value matching "this" host (as determined by comparing with the [ldap_local_host](#) and [ldap_host_alias_list](#) MTA options' value), or has no `mailHost` attribute at all (common for groups and lists), or has `mailDeliveryOption` clauses which are all "mailhost-independent", then its `mailDeliveryOption` value(s) (more precisely, the values of whatever LDAP attribute is named by the [ldap_delivery_option](#) MTA option) will be interpreted as specified by the [delivery_options](#) MTA option, and any appropriate address changes or forwarding will be applied. For instance, a recipient address that is found to correspond to a local user LDAP entry with a `mailDeliveryOption` value of `mailbox` will be converted to the proper local mailbox address as defined by [delivery_options](#); and a recipient address that is found to have a `mailDeliveryOption` value of `forward` will (in accordance with [delivery_options](#)) be converted to any specified `mailForwardingAddress` value(s) (more precisely, be converted to values of the LDAP attribute named by the [ldap_forwarding_address](#) MTA option).
 5. *Configure canonicalization of all non-envelope-To occurrences of "local" addresses.* All non-envelope-To addresses (all addresses which didn't meet the `$$$F` criteria of the rewrite rule in step (2)) are looked up in LDAP via the [reverse_url](#) MTA option template to determine whether (and if so, how) the address should be "reversed" (canonicalized) to some preferred form.

A great many refinements, adjustments, and further optional processing can be configured for the MTA---see especially the various LDAP attribute semantics supported by the MTA listed in [Direct LDAP attribute name MTA options](#) as some modify address handling, as well as further MTA options discussed in [Direct LDAP attribute interpretation MTA options](#), [Direct LDAP usergroup lookup MTA options](#), and [Direct LDAP domain lookup MTA options](#)---so the above steps provide merely a somewhat over-simplified description of the *major* components of Direct LDAP configuration. Note that combinations of Direct LDAP address handling and

more traditional MTA aliasing (via [alias file](#), [alias database](#), or Unified Configuration [alias options](#)) are also possible.

35.2 Aliases in LDAP

If at least one of the [alias_url0](#), [alias_url1](#), [alias_url2](#), or [alias_url3](#) MTA options is specified, then for each address matching the local channel (or any channel marked [aliaslocal](#)) the MTA will automatically perform the LDAP query specified by the [alias_urlN](#) option(s). (If more than one such option is specified, then queries are normally performed in order beginning with [alias_url0](#) and ending with [alias_url3](#); but see the [alias_magic](#) MTA option and [aliasmagic](#) channel option.)

The LDAP server to query, as well as other basic LDAP query parameters, are controlled by certain MTA options and/or `configutil` parameters (legacy configuration) or Unified Configuration [base options](#) and [PAB options](#); see [Table of Basic configuration settings relevant to alias LDAP lookups](#). The MTA options, if explicitly set, for MTA lookup purposes take precedence over (override) their corresponding `configutil` parameters (legacy configuration) or [base options](#) and [PAB options](#) (Unified Configuration).

Note that the MTA's SMTP AUTH user authentication lookups are done using general SASL library code, also used for IMAP, POP, or MSHTTP user logins (authentication). The SASL code does not use the MTA-specific options, but rather uses the `configutil` parameters or Unified Configuration options.

Note that the `*.imta.*` `configutil` parameters can be expected to become obsolete (be removed) in a future version (JES 5 or later).

Table 35.1 Basic configuration settings relevant to alias LDAP lookups

msconfig base option	configutil parameter	MTA option	Default	Description
ugldaphost	<code>local.ugldaphost</code>	ldap_host	++	LDAP host to which to connect
ugldapport	<code>local.ugldapport</code>	ldap_port	389	LDAP port to which to connect
ldapconnecttimeout	<code>local.ldapconnecttimeout</code>		10	Time out, in seconds, for LDAP connections
ldapsearchtimeout+	<code>local.ldapsearchtimeout+</code>	ldap_timeout	180000	Time out, in seconds, for LDAP queries
ugldapbinddn	<code>local.ugldapbinddn</code>	ldap_username		The username with which to bind when doing LDAP queries
ugldapbindcred	<code>local.ugldapbindcred</code>	ldap_password		The password with which to bind when doing LDAP queries
ugldapusessl	<code>local.ugldapusessl</code>		no	Whether to use SSL (LDAP-S) to connect to the user/group LDAP directory; also controls use of SSL for LDAP PAB lookups, and "external" LDAP (<code>extldap</code>) lookups. If set to yes, then the MTA will expect to find a certificate located either in the <code>local.sldbpath</code> directory, if it is specified, or in the MTA's config directory (located as <code>local.instancedir/config</code> , rather than via the usual <code>IMTA_TABLE</code> option value), named <code>local.sldbprefix.secmod.db</code> . In that directory also must be the password file named <code>sslpassword.conf</code> .
ldaprequiretls			0	(New in 8.0) If SSL is not already being used on a given LDAP connection (e.g., due to ugldapusessl being set or use of an <code>ldaps:</code> URL), then enabling <code>base.ldaprequiretls</code> will require successful negotiation of TLS (using LDAP StartTLS) before proceeding with the connection.
Not used as of 7.0	<code>local.instancedir</code>			Prefix for where Messaging Server is installed. In particular, <code>/config</code> is appended onto this as the location for where to find the certificate (if SSL is being used; that is, if <code>local.ugldapusessl</code> is set).

Aliases in LDAP

Deleted in 7.0u4 (never used)	service.imta.ldappoolsize		0	The initial number of LDAP connections to start up
		ldap_max_connections	1024	The maximum number of simultaneous LDAP connections to allow using
defaultdomain	service.defaultdomain	ldap_default_domain		The default domain name
dcroot	service.dcroot	ldap_domain_root	o=internet	The base DN for the domain portion of the DIT
ugldapbasedn	local.ugldapbasedn	ldap_user_root	o=isp	The LDAP user root, that is, the base DN for the user and group portion of the DIT
	local.imta.schematag	ldap_schematag	ims50	The tag for the schema in use
		ldap_group_object_classes	<i>Varies with the schema tag</i>	The object classes required for a group; the default depends upon the schema tag; in particular, for the default schema tag of <code>ims50</code> , the default required object classes are <code>inetLocalMailRecipient+inetmailgroup</code>
		ldap_user_object_classes	<i>Varies with the schema tag</i>	The object classes required for a user; the default depends upon the schema tag; in particular, for the default schema tag of <code>ims50</code> , the default required object classes are <code>inetLocalMailRecipient+inetmailuser</code>
	local.imta.mailaliases	ldap_mail_aliases	<i>Varies with schema tag</i>	The attributes in which aliases are stored; the default depends upon the schema tag; in particular, for the default schema tag of <code>ims50</code> , the default alias attributes are <code>mail</code> , <code>mailAlternateAddress</code> , and <code>mailEquivalentAddress</code>
hostname	local.hostname	ldap_local_host		The local host name (official host name for the "I" channel)
	local.imta.hostnamealiases	ldap_host_alias_list		Local host aliases
ldappoolrefreshinterval	local.ldappoolrefreshinterval		35	Time (minutes) LDAP connections are maintained
ldapcheckcert	local.ldapcheckcert		1	Control whether or not to verify the LDAP server's certificate (when SSL is being used, that is, when ugldapusesl is set to 1)
msconfig PAB option	configutil parameter	MTA option	Default	Description
ldaphost	local.service.pab.ldaphost	ldap_pab_host		(New in 7.0)
ldapbinddn	local.service.pab.ldapbinddn	ldap_pab_username		(New in Messaging Server 7.0)
ldappasswd	local.service.pab.ldappasswd	ldap_pab_password		(New in 7.0)
ldapport	local.service.pab.ldapport	ldap_pab_port	ldap_port value	(New in 7.0) If neither <code>pab.ldapport</code> nor <code>local.service.pab.ldapport</code> is set, then the <code>ldap_port</code> value is used.
		ldap_pab_max_connections	1024	(New in 7.0u1) Maximum simultaneous connections to the PAB

+The `ldapsearchtimeout` base option (Unified Configuration) or `local.ldapsearchtimeout` `configutil` parameter (legacy configuration) is a global default for all searches done through the LDAP pool API, including those done by the MTA.

++The MTA option `ldap_host` defaults to the value of the [ugldaphost](#) base option, which in turn defaults, if not set, to the loopback interface.

Compare this [Table of Basic configuration settings relevant to alias LDAP lookups](#) with [Basic configuration settings relevant to domain LDAP lookups](#).

For the [alias_url0](#), [alias_url1](#), [alias_url2](#), or [alias_url3](#) MTA options, standard LDAP URLs as per [RFC 2255](#) must be used, with the following exception and special interpretations:

- The LDAP server and port are typically omitted, and are instead specified via MTA options or `configutil` parameters (legacy configuration) or base options (Unified Configuration), as shown above in [Table of Basic LDAP settings relevant to alias lookups](#). Indeed, prior to Messaging Server 7.0u4, the host and port had to be omitted; as of Messaging Server 7.0u4, specifying the host and port in the URL itself is supported.
- The MTA makes a distinction between a completely omitted attributes field, which as per [RFC 2255](#) means to request the return of *all* attributes, and an attributes field consisting of

the asterisk character, *, which the MTA instead interprets as meaning to request the return of *all known-to-the-MTA attributes*, that is, all attributes specified by [direct LDAP attribute name MTA options](#). This distinction is available since for some directory setups, there may be a noticeable performance difference in LDAP directory response to one type of query (all attributes requested) *vs.* the other type of query (specific, though large, list of attributes requested).

- Also, certain substitution sequences are available, as shown in [Table of LDAP URL substitution sequences](#).

Thus the LDAP URL value for an `alias_urlN` option 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; it might correspond to the organization's top level in the Directory Information Tree. The optional `attributes`, `scope`, and `filter` portions of the URL further refine what information to return. For an alias, the desired `attributes` to specify returning would typically be the `mail` attribute (or some similar attribute). The `scope` may be any of `base` (the default), `one`, or `sub`. And the desired `filter` would typically be based upon the mailbox (local portion) of the incoming addresses.

Note that the usual LDAP URL encoding rules should be followed; see especially [RFC 1738 \(Uniform Resource Locators \(URL\)\)](#) and [RFC 2255 \(LDAP URL Format\)](#).

Substitution sequences, as shown in [Table of LDAP URL substitution sequences](#), are available for use in constructing the LDAP URL.

The LDAP URL, before any substitutions, is limited to 256 characters in length (252 characters in iMS 5.2 and earlier); the substitutions may insert additional material and the length after such substitutions is limited to 1024 characters. Note that the substitution of "known" attributes when asterisk, *, is specified as the attribute to return, is not considered as part of the regular substitution; this substitution is performed at a later step and the length after this "known" attributes substitution is limited to 4096 characters.

For instance, at a Messaging Server site using [direct LDAP mode](#), `alias_url0` is typically set as follows:

```
domain_uplevel=2
alias_url0=ldap:/// $V?*?sub?$R
```

Here the `domain_uplevel=2` setting means that:

- Since bit 0 (value 1) is not set, domain matches must be exact; (*e.g.*, a domain entry in the DC tree for `siroe.com` will not imply that `host.siroe.com` should also be considered a "local" domain).
- Since bit 2 (value 2) is set, then user alias lookups will be performed looking not only for the exact address presented, but also for that address with the domain name replaced by the "canonical" domain name; for instance, if a domain name is an alias for another domain name (see `ldap_domain_attr_alias` in Schema 1 mode or `ldap_attr_domain2_schema2` in Schema 2 mode), then the user alias lookup will be

performed both with the address as originally presented, and with the address with the domain name replaced by the aliased (to) domain name.

The `alias_url0` setting means that the result of a previous `domainMap` lookup will be used as the base for the search (this is the `$V` substitution), and the MTA will look for its standard set of mail alias attributes (the `$R` substitution); see the `ldap_mail_aliases` MTA option.

If a Messaging Server site is using direct LDAP mode with vanity domains enabled, then typical settings are:

```
domain_uplevel=2
alias_url0=ldap:/// $V?*?sub?$R
alias_url1=ldap:/// $B?*?sub?(&(msgVanityDomain=$D)$R)
alias_url2=ldap:/// $1V?*?sub?(mailAlternateAddress=@$D)
domain_match_url=ldap:/// $B?msgVanityDomain?sub?(msgVanityDomain=$D)
```

In addition to the usual settings (see above), notice the additional `alias_url1`, `alias_url2`, and `domain_match_url` settings. Here the `domain_match_url` setting, used to do an extra lookup when doing `domainMap` checking, means that the user tree base will be used as the base for the search (the `$B` substitution), searching for a `msgVanityDomain` attribute that has the value of the domain name of the address being processed; that is, this is enabling the finding of vanity domain names. The `alias_url1` setting means that the user tree base will be used as the base for a search for entries with `msgVanityDomain` attribute equal to the domain name of the address being processed, and with at least one of the entry's standard mail alias attributes (see the `ldap_mail_aliases` MTA option) equal to the address being processed. The `alias_url2` setting, used if neither the `alias_url0` nor `alias_url1` searches resulted in a match, is looking for an entry that is the "catch-all" address for the domain or vanity domain of the address being processed.

35.3 Aliases in Unified Configuration

In Unified Configuration, plain MTA aliases (but see also [Aliases in LDAP](#)) are set as a named set of options under an `alias group`. At the simplest, an alias can be set as:

```
msconfig> set alias:alias-with-quoted-periods.alias_entry address-or-alias
```

e.g.,

```
msconfig> set alias:first\last@subdomain\domain.com.alias_entry "mailbox@mailhost.domain.com"
```

where the *alias-with-quoted-periods* (appearing between `alias:` and `.alias_entry`) is the alias, and the value on the right hand side is an address or alias to which the alias translates.

Note that each period in the alias name itself must be quoted with the backslash character. After the alias name, an unquoted period marks the end of the alias name, and the start of an `alias option`. The `alias_entry` option is required, being the fundamental definition of the alias; additional alias options are optional.

An alias may translate to multiple addresses or aliases, (forming a so-called mail group). For such an alias, use multiple `set` commands, *e.g.,*

```
msconfig> set alias:first\.last@subdomain\.domain\.com.alias_entry "mailbox@mailhost.domain.com"
msconfig> set alias:first\.last@subdomain\.domain\.com.alias_entry "remote-mailbox@remote-domain.com"
```

Optionally, other [alias options](#) in addition to `alias_entry` may be set on an alias. Such optional alias options are especially relevant when defining a [mailing list](#) (which is merely a specially decorated form of alias, from the MTA's point of view), but may also be set on individual user aliases, *e.g.*:

```
msconfig> set alias:first\.last@subdomain\.domain\.com.alias_blocklimit 200
```

As an alternative to using `msconfig set` commands, you may instead issue the command `edit aliases` within `msconfig` to enter an interactive mode of editing aliases as if they were in the legacy configuration [alias file](#); use `:q` to quit interactive editing mode, or `:x` to save your edits and exit interactive editing mode.

35.4 Alias options

Alias options are the Unified Configuration equivalent of what in legacy configuration were termed [Alias file named parameters](#).

The action of those alias options that cause addition of a header, *e.g.*, [alias_deferredalias_importance](#), *etc.*, can be modified by special characters suffixed on the end of the value, as discussed in [Alias header addition modifiers](#).

35.4.1 Alias options: `alias_entry` (alias or address)

The `alias_entry` alias option is the fundamental part of an alias: it specifies a translation address (or alias) for an alias. For an alias that translates to multiple addresses or aliases, multiple `alias_entry` settings may be made. Each value specified as an `alias_entry` may be a fully qualified address, or a short form alias name itself. Each individual `alias_entry` value corresponds to what in legacy configuration would be one address or alias on the right hand side of an alias. The entire set of `alias_entry` values for an alias corresponds to what in legacy configuration would be the list of addresses/aliases on the right hand side of an alias.

Every MTA configuration should include at least a [postmaster alias](#). *E.g.*,

```
msconfig> show alias.*
role.alias:root@default-domain.alias_entry = postmast
role.alias:root@mta-host.alias_entry = postmast
role.alias:postmaster@mta-host.alias_entry = postmast
```

Since the initial configuration creates in LDAP a postmaster group with `postmaster@default-domain` as the primary address (`mail` attribute) and with `postmast@default-domain` as an alias (`mailAlternateAddress` attribute), these alias entries ensure that alternate "postmaster-ish" sorts of addresses, such as for instance `postmaster@mta-host`, will be directed to the postmaster group.

The distinction between `default-domain` and `mta-host` is the distinction between the default domain for users' e-mail addresses, *vs.* the (possibly different -- indeed a different name is recommended) DNS name of the host system itself. For instance:

```
msconfig> show alias.*
```

alias_alternate_recipient
Option

```
role.alias:root@example\.com.alias_entry = postmast
role.alias:root@host\.example\.com.alias_entry = postmast
role.alias:postmaster@host\.example\.com.alias_entry = postmast
```

35.4.2 alias_alternate_recipient Option

(New in MS 8.0.1.) The `alias_alternate_recipient` alias option is used to associate additional alternate recipient addresses with a group or mailing list.

For users defined in LDAP, see the `ldap_alternate_recipient` MTA option.

35.4.3 Alias options: alias_and and alias_or

The `alias_and` and `alias_or` alias options control whether subsequent access control clauses (e.g., `alias_auth_list`, `alias_auth_mapping`, etc.) are ANDed or ORed. They are analogues of the legacy configuration `alias file named parameters` [AND] and [OR], respectively. The default is controlled by the `or_clauses` MTA option---and is AND (for backwards compatibility) by default. For groups and lists defined in LDAP, see also the AND and OR values for the `mgrpBroadcasterPolicy` attribute (or more precisely, the attribute named by the `ldap_auth_policy` MTA option). For a more detailed discussion, see [Mailing list multiple access control interpretation](#).

35.4.4 Alias options: alias_auth_channel and alias_cant_channel

The `alias_auth_channel` alias option is used to specify a source channel or channels that may submit messages to the mailing list. The `alias_cant_channel` alias option is used to specify a source channel or channels that may not submit messages to the mailing list. These alias options are analogues of the legacy configuration `alias file named parameters` [AUTH_CHANNEL] and [CANT_CHANNEL], respectively. The argument for such options should be a (possibly wildcarded) channel name, or a space-separated list of (possibly wildcarded) channel names.

35.4.5 Alias options: alias_auth_list, alias_cant_list, alias_username_auth_list, and alias_username_cant_list

These options are analogues of the `alias file named parameters` [AUTH_LIST], [CANT_LIST], [USERNAME_AUTH_LIST], and [USERNAME_CANT_LIST], respectively.

35.4.6 Alias options: alias_auth_mapping and alias_cant_mapping

These options are analogues of the `alias file named parameters` [AUTH_MAPPING] and [CANT_MAPPING], respectively.

35.4.7 Alias options: alias_auth_username and alias_cant_username

The `alias_auth_username` and `alias_cant_username` alias options are analogues of the [alias file named parameters](#) [AUTH_USERNAME] and [CANT_USERNAME], respectively.

35.4.8 Alias options: `alias_autosecretary`

RESTRICTED: Not yet implemented.

35.4.9 Alias options: `alias_blocklimit` and `alias_linelimit`

The `alias_blocklimit` and `alias_linelimit` alias options may be used to limit the size of messages that may be posted to the address (whether user or group or list). The value must be an integer number of [MTA blocks](#) for `alias_blocklimit`, or an integer number of lines for `alias_linelimit`. The number of bytes in a block is specified via the `block_size` MTA option. By default, neither `alias_blocklimit` nor `alias_linelimit` is set; being unset (or being set to a value of 0) means that they impose no limit on the size of message that may be posted to the address (though other limits, such as channel or system wide limits, may be in effect). In particular, neither `alias_blocklimit` and `alias_linelimit` will override more general limits that may be in effect; they are minimized with any such general limits. For user, groups, and lists defined in LDAP, see `mailMsgMaxBlocks` attribute (or more precisely, the attribute named by the `ldap_blocklimit` MTA option).

The legacy configuration analogues are the [\[BLOCKLIMIT\]](#) and [\[LINELIMIT\]](#) [alias file named parameters](#).

35.4.10 Alias options: `alias_capture` and `alias_journal`

The `alias_capture` alias option may be used to set an address to which to direct an [encapsulated, "captured" copy](#) of each message posted to the list. The `alias_journal` alias option works similarly, but generates an [envelope "journal" format message](#). The value item should be the address to which to send the "captured" message copies. These alias options are exactly analogous to use of the LDAP attribute named by the `ldap_capture` MTA option on a user or group or mailing list defined via an LDAP entry.

The legacy configuration analogues are the [\[CAPTURE\]](#) and [\[JOURNAL\]](#) [alias file named parameters](#).

35.4.11 Alias options: `alias_conversion_tag`

The `alias_conversion_tag` alias option may be used to set a [tag](#) which conversion file entries can match upon. The value item should be the string to use as the tag. For instance, if a list is defined

```
msconfig> show alias:testlist@domain\.com.*
role.alias:testlist@domain\.com.alias_entry = user1@domain.com
role.alias:testlist@domain\.com.alias_entry = user2@domain.com
role.alias:testlist@domain\.com.alias_entry = remoteuser@remote.com
role.alias:testlist@domain\.com.conversion_tag = listtag
role.alias:testlist@domain\.com.envelope_from = user2@domain.com
```

alias_creation_date alias
option

then conversion file entries could include a `tag=listtag;` clause to match. For instance, if for some mailing list it was desired to convert any text/html parts in posted messages to text/plain, and if a site had an HTML to TEXT convertor called `htmltotextconvert` stored in the `IMTA_PROGRAM` directory (`data-root/site-programs/`), and had set up the conversion channel and a [CONVERSIONS mapping table](#) to apply to list postings, then a conversion file entry could be

```
in-chan=*; out-chan=*; in-type=text; in-subtype=html; tag=listtag;
out-type=text; out-subtype=plain; parameter-copy-0=*;
command="IMTA_PROGRAM:htmltotextconvert $INPUT_FILE $OUTPUT_FILE"
```

For users, groups, and lists defined in LDAP, see the `mailConversionTag` attribute (or more precisely, the attribute named by the `ldap_conversion_tag` MTA option).

In legacy configuration, the analogue is the [\[CONVERSION_TAG\] alias file named parameter](#).

35.4.12 Alias options: `alias_creation_date`

(New in 8.0.) The `alias_creation_date` alias option may be used to set a creation date for the alias (intended to be used for RRV5 purposes). The creation date value must be in [RFC 3339](#) (Date and Time on the Internet: Timestamps) format (a profile of [ISO 8601 format](#)), along the lines of:

```
YYYY-MM-DDTHH:MM:SS.ssZ
```

or

```
YYYY-MM-DDTHH:MM:SS.ssplus-or-minusHH:MM
```

where the hundredths of seconds portion is optional, and T and (if used) Z are not case sensitive. For instance:

```
2014-02-28T12:13:14.30-07:00
```

This alias option is analogous to use of the LDAP attribute named by the [ldap_creation_date](#) MTA option on a user or group or mailing list defined via an LDAP entry, or to use of the LDAP attribute named by the [ldap_domain_attr_creation_date](#) MTA option on a domain entry.

The legacy configuration analogue is the [\[CREATION_DATE\] alias file named parameter](#).

35.4.13 Alias options: `alias_deferred` (ISO 8601 P time duration string), `alias_deferred_list` (filepath or MTA URL), `alias_deferred_mapping` (MTA mapping name)

The `alias_deferred` alias option may be used to add a Deferred-delivery: header line. The value should be a date and time, in [ISO 8601 P format](#).

The `alias_deferred_list` alias option takes two values, a file specification for a file containing a list of originator addresses (or a URL returning a list of originator addresses) to

whose postings to add a Deferred-delivery: header, and the deferral date/time in [ISO 8601 format](#).

As of 8.0 (in prior versions, this feature "existed" but was not working), the `alias_deferred_mapping` alias option may be used to specify a mapping table through which to run originator addresses. The `alias_deferred_mapping` alias option takes one or two arguments, with a space between if the optional second argument is included. The first argument is required and must contain at a minimum the name of an MTA mapping table; the first argument may also, optionally, include a vertical bar followed by a string to use as a prefix in the mapping table probe, prior to the originator address. The second argument is optional, consisting of a deferral date/time in [ISO 8601 format](#).

```
mapping-name[|probe-prefix] ISO-8601-deferral-time
```

Originator addresses will be run through the specified mapping. If the mapping template does not begin with an N, n, F, or f, and if it contains a valid date/time specification in ISO 8601 format, then that date/time will be used as a deferral time. If the mapping template does not contain a date/time specification yet does not begin with N, n, F, or f, then the deferral date/time specified as the second argument to `alias_deferred_mapping` will be used. The default, if no mapping entry matches, or if an entry that begins with an N, n, F, or f matches, is not to add a Deferred-delivery: header. Note that the intended purpose of a `probe-prefix` is for convenience in using a single MTA mapping table for multiple mailing list deferral settings, *e.g.*, by using a probe prefix consisting of the list name, so that entries in the mapping table may be list specific. Similarly, a deferral time specified as the second argument permits a default deferral time, that may then be overridden in the case of specific originators in the mapping table result.

Setting bit 3/value 8 of the `include_connectioninfo` MTA option will cause additional information to be included in the input probe of the mapping named by `alias_deferred_mapping`. Thus if a `probe-prefix` has also been specified, then the probe will take the form:

```
transport-info|application-info|probe-prefix|originator-address
```

Note that by default the MTA does not honor Deferred-delivery: headers; see the [deferreddestination channel option](#) for a discussion. As a functionally preferable alternative to the Deferred-delivery: header line approach for retaining/deferring messages, see also the [SMTP SUBMIT FUTURERELEASE extension](#).

35.4.14 Alias options: `alias_delay_notifications` and `alias_nodelay_notifications`

The `alias_delay_notifications` and `alias_nodelay_notifications` alias options are analogues of the [\[DELAY_NOTIFICATIONS\]](#) and [\[NODELAY_NOTIFICATIONS\]](#) alias file named parameters. The `alias_delay_notifications` alias option requests that NOTARY delay notifications be sent for mailing list postings; the `alias_nodelay_notifications` alias option requests that NOTARY delay notifications not be sent for mailing list postings.

For lists defined in LDAP, the analogous settings are controlled via whatever attribute is named by the `ldap_delay_notifications` MTA option, by default `mgrpDelayNotifications`.

35.4.15 Alias options: `alias_digest_recurrence`

alias_direct_list and
alias_direct_mapping alias
options

RESTRICTED: Not yet fully implemented.

The `alias_digest_recurrence` alias option takes an [ISO 8601](#) argument.

35.4.16 Alias options: `alias_direct_list` and `alias_direct_mapping`

RESTRICTED: Not yet fully implemented.

`alias_direct_list` takes a file specification for a file containing a list of originator addresses (or a URL returning a list of originator addresses).

`alias_direct_mapping` takes the name of a mapping table through which to run originator addresses.

35.4.17 Alias options: `alias_envelope_from`

The `alias_envelope_from` alias option takes a required value specifying an address to replace the message's original envelope From address. The legacy configuration analogue is the [\[ENVELOPE_FROM\] alias file named parameter](#). This sets only the envelope From address, (unlike the `alias_file_error_return_address` positional parameter which also sets an `Errors-to: address`).

Setting the value to an address of the form `user+*@domain` has a special meaning. The asterisk character will be expanded into a representation of the recipient address; thus a separate copy of the list message is generated for each recipient, with each copy including the intended recipient address as a [subaddress](#) within the return address. If delivery errors subsequently occur, the subaddress will indicate which was the failing address. In some cases, when dealing with remote MTAs that generate nonstandard, uninformative delivery error messages, this can in theory be useful as a way of determining which recipient address(es) failed, even when the bounce message's inner content is relatively uninformative. And it may make processing of such bounce messages by an automated program more convenient. However, the tradeoff is that such per-user-specific return address values require that a separate message copy be generated and sent for each recipient; for a "large" list, with many recipients in the same destination domains, this can be a large increase in overhead (a large decrease in efficiency). And with more prevalent use nowadays of standard format [notification messages](#), the "need" for this sort of approach, with its extra (potentially large) overhead, is much less (since the intended recipient information can instead be extracted from the standard field in the contents of a standard format notification message).

(New in JES MS 6.3.) Setting the value to the forward slash character, `/`, has a special meaning. It tells the MTA to revert to using the original envelope From address that had been present on the incoming message, yet in all other respects use mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender.

For groups and lists defined in LDAP, see the `mgrpErrorsTo` attribute (or more precisely, the attribute named by the `ldap_errors_to` MTA option).

35.4.18 Alias options: `alias_error_text`

The `alias_error_text` alias option specifies a string to use as the "reason" which will be returned to the attempted sender if and when an attempted posting fails due to an access failure, overriding the usual error text that would be returned in such a case. For groups

defined in LDAP, see the `mgrpRejectText` attribute or `mgrpMsgRejectText` attribute (or more precisely, whatever attribute(s) are named by the `ldap_reject_text` MTA option).

For aliases defined in the alias file or alias database, the analogous setting is the `[ERROR_TEXT]` alias file named parameter.

35.4.19 Alias options: `alias_expandable` and `alias_nonexpandable`

The `alias_expandable` alias option (analogue of the legacy configuration alias file named parameter `[EXPANDABLE]`) is used to specify that the associated list can be expanded (and hence its contents seen) by various protocols which may attempt such an operation. It does not mean, or imply, that the membership of the list will be expanded into message headers; for that, instead see [alias_header_expansion](#).

The `alias_nonexpandable` alias option (analogue of the legacy configuration alias file named parameter `[NONEXPANDABLE]`) specifies that the associated list may not be expanded.

`alias_expandable` is the default, unless the `expandable_default` MTA option has been set, in which case the default is `alias_nonexpandable`.

`alias_nonexpandable` is useful in blocking the expansion of mailing lists via SMTP's EXPN command. Note that mailing list access controls, *e.g.*, [alias_auth_list](#), [alias_auth_mapping](#), *etc.*, also affect the expansion of mailing lists via SMTP's EXPN command; the SMTP server will only permit the EXPN if the SMTP client passes the access control (*e.g.*, has issued a prior MAIL FROM: command that passes the access control).

35.4.20 Alias options: `alias_expiry`

The `alias_expiry` alias option is used to add an Expiry-date: header line. The value should be a date and time, in [ISO 8601 P format](#). (The MTA will convert the specified value into the appropriate corresponding [RFC 5322](#) date value needed for the header line.) The MTA's periodic return job will return messages whose Expiry-date: has passed.

35.4.21 Alias options: `alias_filter`

The `alias_filter` alias option is the analogue of the legacy configuration [\[FILTER\] alias file named parameter](#). It takes a URL argument specifying the location of a Sieve filter to apply on attempted message postings. The argument may be any [supported form of URL](#) that makes sense; in particular, besides supporting `file:file-spec` URLs or simply file specifications without the leading `file:`, LDAP URLs, and `data:sieve-commands` are also supported. Note that when specifying a file, it must be the full file specification for the filter file to apply.

35.4.22 Alias options: `alias_header_addition` and `alias_header_trim`

The `alias_header_trim` and `alias_header_addition` alias options are analogues of the legacy configuration [\[HEADER_TRIM\]](#) and [\[HEADER_ADDITION\]](#) [alias file named parameters](#). The `alias_header_trim` alias option may be used to add headers to or remove headers from posted messages. The argument must be a full file specification for a header

trimming option file; see [Header option files](#) for information on the format of these files. `alias_header_addition` is more specialized than `alias_header_trim`, being used when there are merely headers to be added. `alias_header_addition` may be used to specify a file of headers to be added to posted messages. The argument must be a full file specification for the file containing headers to be added.

In particular this facility can be used to add the standard mailing list headers defined in [RFC 2369](#). For instance, a site `domain.com` that has set up a list named `listname`, that has a list owner address of `listname-owner@domain.com` and a list members administrator address of `listname-request@domain.com`, and with certain list information and archives available at an FTP site, might use a header addition file along the lines of the following:

```
List-Help: <ftp://ftp.domain.com/pub/listname-help.txt> (FTP),
<mailto:listname-owner@domain.com?subject=help> (List Manager)
List-Subscribe:
<mailto:listname-request@domain.com?subject=subscribe%20listname?body=subscribe%20listname>
List-Unsubscribe:
<mailto:listname-request@domain.com?subject=unsubscribe%20listname?body=unsubscribe%20listname>
List-Post: <mailto:listname@domain.com>
List-Owner: <mailto:listname-owner@domain.com?Subject=listname>
List-Archive: <ftp://ftp.domain.com/pub/listname/archive/>,
<mailto:listname-request@domain.com?subject="send%20listname%20archives?body=send%20/pub/listname/archive/*">
```

35.4.23 Alias options: `alias_header_alias` and `alias_header_expansion`

The `alias_header_alias` and `alias_header_expansion` alias options are Unified Configuration analogues of the [\[HEADER_ALIAS\]](#) and [\[HEADER_EXPANSION\]](#) alias file named parameters. Because their effect is limited to cases such as messages submitted via the L channel, they have almost no relevance in modern Messaging Server configuration.

35.4.24 Alias options: `alias_header_check`

(New in Messaging Server 7.0.5.) The `alias_header_check` alias option is used in conjunction with a `addrtypescan*` channel option. Valid arguments are `jettison` or `discard`. This alias option is an analogue of the LDAP attribute named by the [ldap_check_header](#) MTA option.

35.4.25 Alias options: `alias_hold_list`, `alias_nohold_list`, `alias_hold_mapping`, `alias_nohold_mapping`

The `alias_hold_list` alias option may be used to specify a list of originator addresses whose attempts to post to the list should be sidelined as `.HELD` messages. The argument may be any [supported form of URL](#) that makes sense. The `alias_nohold_list` alias option may be used to specify the list of originator addresses whose postings should not be so sidelined, while all other postings will be sidelined. The value must be a full file specification for a file of addresses, or an LDAP URL returning a list of addresses. The `alias_hold_mapping` and `alias_nohold_mapping` alias options are used analogously, but via [mapping tables](#) rather than via lists. The value should be the name of an MTA mapping table.

These alias options are analogues of the alias file/alias database named parameters `HOLD_LIST`, `NOHOLD_LIST`, `HOLD_MAPPING` and `NOHOLD_MAPPING`.

alias_importance,
alias_precedence,
alias_priority, and
alias_sensitivity alias
options

35.4.26 Alias options: `alias_importance`, `alias_precedence`, `alias_priority`, and `alias_keep_read`

The `alias_importance`, `alias_precedence`, `alias_priority`, and `alias_sensitivity` alias options are used to generate respective header lines; the value specified is inserted on the respective header line. Alias file/alias database analogues exist; see the [\[IMPORTANCE\]](#), [\[PRECEDENCE\]](#), [\[PRIORITY\]](#), and [\[SENSITIVITY\]](#) alias file named parameters.

Note that the more general [alias_header_addition](#) alias option -- in legacy configuration, the alias file/alias database [\[HEADER_ADDITION\]](#) named parameter -- provides an alternate way to add these and other header lines. Or for aliases defined in LDAP, see the [ldap_add_header](#) MTA option.

35.4.27 Alias options: `alias_keep_delivery` and `alias_keep_read`

The `alias_keep_delivery` and `alias_keep_read` alias options are Unified Configuration analogues of the alias file named parameters [\[KEEP_DELIVERY\]](#) and [\[KEEP_READ\]](#).

35.4.28 Alias options: `alias_list_name`

New in Messaging Server 7.4-0.01. RESTRICTED: Reserved for future use.

35.4.29 Alias options: `alias_moderator_address`, `alias_moderator_list`, `alias_moderator_mapping`, `alias_username_moderator_list`

The `alias_moderator_*` alias options are used to establish a [moderated mailing list](#). All postings to the list not originating from a moderator are sent to the list's moderator. The address of the moderator must be specified with the `alias_moderator_address` alias option. The moderator address determines where moderator mail is sent when someone other than the moderator posts. The value of that named parameter is the moderator's address. For example,

```
msconfig> show alias:test-list@domain\.com
instance.alias:test-list@domain\.com.alias_entry = <IMTA_TABLE:test.dis
instance.alias:test-list@domain\.com.alias_moderator_address = bob@domain.com
```

When there may be multiple moderator addresses (for instance, both `robert@mail1.domain.com` and `bob@domain.com`), use `alias_moderator_list`, `alias_username_moderator_list`, or `alias_moderator_mapping` to specify all addresses from which postings should be passed directly to the list and not sent to the list's moderator.

`alias_moderator_list` specifies either the name of a file containing a list of moderator addresses, or an LDAP URL returning a list of moderator addresses.

alias_*originator_reply
alias options

`alias_username_moderator_list` specifies as its argument a [URL that "makes sense"](#): either the name of a file containing a list of (possibly wildcarded) moderator usernames, or an LDAP URL returning a list of (possibly wildcarded) moderator usernames; note that usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting MTA process is running. Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, *, character. For instance, to specify that only the user JDOE is the list moderator, whether submitting from the L channel or via SMTP (e.g., from a POP or IMAP client that performs SASL SMTP authentication), the `alias_username_moderator_list` file would need to contain the entries:

```
JDOE
$*JDOE
```

where the first entry would match for messages submitted from the L channel and the second entry would match for messages submitted via SMTP AUTH. Note that as asterisk is normally a wildcard character, matching of only the exact literal asterisk character is specified by using the dollar character to quote the asterisk.

`alias_moderator_mapping` specifies the name of a [mapping table](#) used to verify whether or not an address is a moderator address.

See also the `alias_sasl_moderator_list` and `alias_sasl_moderator_mapping` alias options, which operate similarly but *require* that an authenticated address be present in the attempted posting.

If an `alias_moderator_list`, `alias_moderator_mapping`, `alias_sasl_moderator_list`, or `alias_sasl_moderator_mapping` alias option is used, thereby specifying who may post directly to the list, then an `alias_moderator_address` alias option should also be present to specify the address to which to send postings not from any moderator.

The use of the `alias_moderator_address` alias option alone, without the `alias_moderator_list` alias option, is equivalent to using `alias_moderator_address` and an `alias_moderator_list` consisting of just the one moderator address.

Legacy configuration has analogous [named parameters](#) `[MODERATOR_ADDRESS]`, `[MODERATOR_LIST]`, `[MODERATOR_MAPPING]`, and `[USERNAME_MODERATOR_LIST]`, as well as [named parameters](#) `[SASL_MODERATOR_LIST]` and `[SASL_MODERATOR_MAPPING]`.

For lists defined in LDAP, the configuration of moderation is structured somewhat differently; see the LDAP attributes named by the `ldap_reject_action`, `ldap_moderator_url`, and `ldap_auth_url` MTA options.

35.4.30 Alias options: `alias_originator_reply`, `alias_nooriginator_reply`

The `alias_originator_reply` alias option is used to control whether or not the originator's address is added to any generated Reply-to: header. The `value` item should be the [full file path specification for a world readable file, or a resolvable URL](#), containing the list of addresses that should never be added. (This is usually the mailing list itself.) The MTA

alias_received*,
alias_noreceived* alias
options

will match the envelope From address against the addresses in the list; if no match occurs, the originator's address will be added to any generated Reply-to: header.

alias_nooriginator_reply specifies that any generated Reply-to: header should contain only explicitly specified addresses. A value is required, but ignored.

If neither alias_originator_reply nor alias_nooriginator_reply is explicitly set, the MTA's default behavior is effectively equivalent to alias_nooriginator_reply.

In legacy configuration, the analogous alias file named parameters are [NOORIGINATOR_REPLY](#) and [ORIGINATOR_REPLY](#).

35.4.31 Alias options: `alias_receivedfor`, `alias_noreceivedfor`, `alias_receivedfrom`, `alias_noreceivedfrom`

The `alias_receivedfor`, `alias_noreceivedfor`, `alias_receivedfrom`, and `alias_noreceivedfrom` alias options control features of what appears in the Received: header constructed when expanding the alias, and override normal channel [receivedfor](#), [noreceivedfor](#), [receivedfrom](#), or [noreceivedfrom](#) channel option settings. The value specification is currently ignored and should always be NONE.

In legacy configuration, the analogues for these alias options are the alias file named parameters [\[RECEIVEDFOR\]](#), [\[NORECEIVEDFOR\]](#), [\[RECEIVEDFROM\]](#), and [\[NORECEIVEDFROM\]](#).

35.4.32 Alias options: `alias_nosolicit`

The `alias_nosolicit` alias option sets a solicitation keyword, or a comma-separated list of solicitation keywords, that will not be allowed on postings to the list. Attempted postings to an alias address that has such a keyword set will be rejected with SMTP error:

550 5.7.1 Solicitation check failure on SOLICIT=*solicitation-keyword*: *recipient-address*

In legacy configuration, the analogous alias file named parameter is [\[NOSOLICIT\]](#).

35.4.33 Alias options: `alias_optin`

The legacy configuration `alias_optin` alias option has been replaced in Unified Configuration by [alias_optin1](#).

35.4.34 Spam/virus filter "opt in" alias options: `alias_optinN (string)`

The `alias_optinN` alias options each respectively set an opt-in value for "opting in" to [spam/virus filter package N](#), for N in the range 1 to 8. (`alias_optin` is a synonym for `alias_optin1`.)

These alias options are analogues of the legacy configuration alias file named parameters [\[OPTINn\]](#).

See also the [ldap_opt in MTA options](#) for similar functionality for aliases stored in LDAP.

35.4.35 Password protection for postings: alias_password (string)

The `alias_password` alias option specifies a password, or a comma-separated list of passwords, that allow posting to the list. An attempted posting to the list must contain one of these values on an `Approved:` header line in order for the posting to be allowed. During mailing list expansion, the password value will be removed from the `Approved:` header line; indeed, if that is the only value on the `Approved:` header line, then the entire header line will be removed. See [Password-protected mailing lists](#).

In legacy configuration, the analogous alias file named parameter is `[PASSWORD]`.

For aliases/lists defined in LDAP, the analogous setting is the LDAP attribute named by the `ldap_auth_password` MTA option.

35.4.36 Disclaimer/text addition alias options: alias_prefix_text (string), alias_suffix_text (string)

The `alias_prefix_text` and `alias_suffix_text` alias options cause insertion of, respectively, prefix or suffix text into messages as they undergo list expansion. Prior to Messaging Server 7.3-11.01, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.3-11.01, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The value (the text) is specified in UTF-8; this is then converted to match the charset of the part into which the text is being inserted.

In legacy configuration, the analogous alias file named parameters are `[PREFIX_TEXT]` and `[SUFFIX_TEXT]`.

For aliases/lists defined in LDAP, see the `ldap_prefix_text` and `ldap_suffix_text` MTA options.

Or see the [Sieve addprefix and addsuffix extensions](#).

35.4.37 Alias options: alias_presence

RESTRICTED: Not yet implemented.

35.4.38 Alias options: alias_private and alias_public

The `alias_private` and `alias_public` alias options are analogues of the `[PRIVATE]` and `[PUBLIC]` alias file named parameters. The `alias_public` alias option specifies that the associated alias is public and hence can appear in any constructed header lines. The value specification is currently ignored and should always be `NONE`. `alias_public` is the default. The `alias_private` alias option specifies that the alias is private and should appear as an empty group construct in message headers. The value specification is used as

the name for the group. Neither `alias_public` nor `alias_private` have any effect if the `alias_header_expansion` alias option is also specified.

Note that these parameters are only valid when headers are originally being constructed, as for instance for messages submitted via the L channel. These parameters are not relevant for incoming messages (such as incoming SMTP messages) for which the headers are already present in one form or another.

35.4.39 Deferred expansion alias option: `alias_reprocess` (string)

The `alias_reprocess` alias option is used to request deferred expansion of the mailing list, where rather than expanding the mailing list "on line", the message should instead be enqueued to the `reprocess channel`; the reprocess channel can then perform the mailing list processing in a separate step. The value specification is currently ignored and should always be `reprocess`.

Use of this alias option defers much of the processing overhead of handling the message to the later step when the reprocess channel runs, rather than doing the processing as the message is initially accepted. This deferred processing can be especially helpful in cases such as incoming SMTP messages addressed to large mailing lists, where "on line" delays could lead to connection time outs.

Use of this parameter as in:

```
listname: </pmdf/table/listname.dis, [REPROCESS] reprocess
```

thus provides essentially identical functionality as defining a mailing list in two stages through the reprocess channel to obtain deferred expansion (the mailing list addresses aren't even expanded until the reprocess channel runs) such as:

```
listname: listname-expand@reprocess
listname-expand: </pmdf/table/listname.dis
```

In legacy configuration, the analogous alias file named parameter is `[REPROCESS]`.

For aliases/lists defined in LDAP, see the `ldap_reprocess` MTA option.

35.4.40 SASL-based access alias options: `alias_sasl_auth_list` (file or URL), `alias_sasl_auth_mapping` (MTA mapping name), `alias_sasl_cant_list` (file or URL), `alias_sasl_cant_mapping` (MTA mapping name), `alias_sasl_moderator_list` (file or URL), `alias_sasl_moderator_mapping` (MTA mapping name)

The `alias_sasl_*` alias options are analogues of the [non-SASL similar alias options](#), but with the additional requirement that an authenticated address be present in the message (whether that be an address literally authenticated via SMTP AUTH, or forced via, *e.g.*, an [authrewrite](#) or [FROM_ACCESS](#) effect).

In legacy configuration, they have analogues in the alias file named parameters [\[SASL_AUTH_LIST\]](#), [\[SASL_AUTH_MAPPING\]](#), [\[SASL_CANT_LIST\]](#), [\[SASL_CANT_MAPPING\]](#), [\[SASL_MODERATOR_LIST\]](#), [\[SASL_MODERATOR_MAPPING\]](#).

35.4.41 Alias options: `alias_sequence_prefix` (file-path), `alias_sequence_suffix` (file-path), `alias_sequence_strip` (string)

The `alias_sequence_prefix` and `alias_sequence_suffix` alias options request that a sequence number be prepended or appended to the Subject: lines of messages posted to the list. They are analogues of the legacy alias file named parameters [\[SEQUENCE_PREFIX\]](#) and [\[SEQUENCE_SUFFIX\]](#). The `value` item gives the full file path specification of a sequence number file. This file is read, incremented, and updated each time a message is posted to the list. The number read from the file is prepended, in the case of `alias_sequence_prefix`, or appended, in the case of `alias_sequence_suffix`, to the message's Subject: header line. This mechanism provides a way of uniquely sequencing each message posted to a list so that recipients can more easily track postings and determine whether or not they have missed any.

By default, a response to a previously posted message (with a previous sequence number) retains the previous sequence number as well as adding a new sequence number to the subject line; the build up of sequence numbers shows the entire "thread" of the message in question. However, the `alias_sequence_strip` alias option (analogue of the [alias file named parameter \[SEQUENCE_STRIP\]](#)) can be used to request that only the highest numbered, *i.e.*, most recent, sequence number be retained on the subject line. The `value` item is currently ignored and should always be `NONE`.

Important note: To ensure that sequence numbers are only incremented for successful postings, an `alias_sequence_prefix` or `alias_sequence_suffix` alias option should always be set as the last alias option; that is, if other alias options are also being used, the `alias_sequence_*` options should be set (and appear when shown) at the end of the list of alias options on an alias entry.

Sequence number files are binary files and must have the proper file attributes and access permissions in order to function correctly.

35.4.42 Per-recipient message copy alias option: `alias_single` (string)

The `alias_single` alias option, if set, forces a separate message copy per recipient (per list member). Thus it can be considered a per-list analogue of the [single](#) channel option. It takes string argument, currently ignored, which should be set to the value `"NONE"`.

In legacy configuration, its analogue is the alias file named parameter [\[SINGLE\]](#).

35.4.43 Extra value alias options: `alias_spare*` (string)

The `alias_spareN` alias options ($N = 1, \dots, 18$) are analogous to the attributes named by the `ldap_spare_N` MTA options. In legacy configuration, the analogous alias file named parameters are `[SPARE*]`.

35.4.44 Tag inserted on Subject: header line alias option: `alias_tag` (string)

The `alias_tag` alias option may be used to prefix specified text to the Subject: header of posted messages. The value should be the string to be added. The string should not contain the vertical bar, `|`, character; prior to JES MS 6.3, the string should not have contained the space character. For instance,

```
msconfig> set alias:schedule-list@domain\.com.alias_tag "Schedule posting -- "
msconfig# show alias:schedule-list@domain\.com
instance.alias:schedule-list@domain\.com.alias_entry = "<ldap:///o=usergroup?mail?sub?(isMember=schedule-list)"
instance.alias:schedule-list@domain\.com.alias_auth_list = "<ldap:///o=usergroup?mail?sub?(isMember=schedule-list)"
instance.alias:schedule-list@domain\.com.alias_tag = "Schedule posting -- "
```

will cause any postings to the list `schedule-list` to have a Subject: header that begins "Schedule posting -- " followed by whatever the original subject of the posting might have been. See the `ldap_add_tag` MTA option for setting an attribute name to provide analogous functionality for lists defined in LDAP, and in legacy configuration, see instead the alias file named parameter `[TAG]`.

35.4.45 To: header line alias option: `alias_to` (string)

The `alias_to` alias option specifies what to put on the To: header line of postings to the mailing list.

In legacy configuration, see the alias file named parameter `[TO]`.

35.4.46 Alias options: `alias_username` (string)

The `alias_username` alias option may be used to set the "username" that the MTA will consider to "own" these mailing list messages. (The legacy configuration alias file named parameter equivalent is `[USERNAME]`.) The `imsimta` `qm` utility will allow that username to inspect and bounce messages in the queue resulting from expansion of this mailing list. The value item should be the username of the account to "own" the mailing list postings.

35.5 Alias file

In legacy configuration, especially in older MTA configurations, aliases were stored in the MTA alias file, normally named `aliases` and stored in the MTA table (`config`) directory. In more modern MTA configurations, most [aliases are stored instead in LDAP](#), with only a few basic aliases stored elsewhere: either in the `aliases` file in legacy configuration, or as a set of values in an `alias` group in Unified Configuration.

Even though in Unified Configuration the old `aliases` file is not actually used, the `alias` settings may be viewed "as if" they were in the `aliases` file by using the `msconfig` command `edit aliases`:

```
msconfig> edit aliases
```

35.5.1 Alias file format

The alias file format is as follows:

```
alias1: a1,a2,...,am
alias2: b1,b2,...,bm
.      .
.      .
.      .
aliasn: n1,n2,...,nm
.      .
.      .
.      .
```

where `aliasn` is translated into the addresses `n1, n2, n3, ..., nm`. The aliases `alias1, alias2, ..., aliasn` are limited to 128 characters each. (In iMS 5.2 and earlier, the limit had been 64 characters.) Each address `a1, a2, etc.`, may contain up to 256 characters (252 characters in iMS 5.2 and earlier). There is no limit to the number of addresses that can be specified for an alias (that is, appear in a single list on the right hand side of an alias definition), although excessive numbers of addresses may eat up excessive amounts of memory. A physical line of the alias file may contain at most 1024 characters. To specify a list of addresses containing more than that number of characters, the line must be continued onto multiple physical lines. Long lines may be continued by ending them with a backslash, `\`. A backslash must follow a comma. There can be no white space preceding the colon separating the alias name from its translation value.

An alias expansion address prefixed with a colon character, `:`, has a special interpretation. The alias expansion address will be used as normal *except* when the MTA is generating a [notification message](#) (a Delivery Status Notification such as a bounce message, or a Message Disposition Notification such as a vacation messages); when generating a notification message regarding the alias, the unexpanded alias will be used rather than the alias expansion value (which would normally be used). This mechanism can be useful in cases where an alias expansion address is an "internal" address that should not be exposed to the outside. It is essentially an alias-specific analogue of the [useintermediate](#) channel option. For instance, with aliases defined as

```
adam: :bob@ims-ms-daemon
carl: donald@ims-ms-daemon
```

messages sent to `adam` will be redirected to `bob@ims-ms-daemon`. But if a notification message needs to be sent back to an original message sender, the notification message will refer to address `adam`, rather than to `bob@ims-ms-daemon`. This contrasts with the case of notification messages regarding messages sent to `carl`; in this case, any notification messages will refer to the `donald@ims-ms-daemon` address.

The matching process is configurable for aliases containing a subaddress, that is, aliases of a form such as:

```
adam+hobbylist: adam-personal-mailbox@domain.com
```

See the [subaddress*](#) channel options for details.

An address (or addresses) on the right hand side of an alias file entry may optionally have various so-called [named parameters](#) associated with it (or them). Such named parameters are more commonly of interest and used with [mailing list definitions](#), but some (in particular, [BLOCKLIMIT], [CAPTURE], [JOURNAL], [CONVERSION_TAG], and [FILTER]), can be of interest for individual aliases as well. Such parameters are specified by listing them at the *beginning* of the right hand (translation) side of the alias entry; the parameters then apply to all addresses on the right hand. Thus an alias entry using named parameters and translating to a single addresses would have the form:

```
alias: [p-name-1] p-value-1, ..., [p-name-k] p-value-k, address
```

while an alias entry using named parameters and translating to multiple addresses (hence an e-mail group, or perhaps mailing list, depending upon which named parameters are set) would have the form:

```
alias: [p-name-1] p-value-1, ..., [p-name-k] p-value-k, address-1, ..., address-j
```

corresponding to Unified Configuration [alias option](#) settings along the lines of:

```
msconfig> show alias:alias
alias:alias.alias_entry = address-1
alias:alias.alias_entry = address-2
...
alias:alias.alias_entry = address-j
alias:alias.p-name-1 = p-value-1
alias:alias.p-name-2 = p-value-2
...
alias:alias.p-name-k = p-value-k
```

Alternatively, rather than having an address or (in legacy configuration) a comma separated list of addresses as the translation of an alias, in the alias file an alias may translate to a mailing list reference as discussed in [Alias file mailing list aliases](#), or to an LDAP URL reference as discussed in [Alias file LDAP URL alias values](#).

A typical, minimal alias file will include at least a postmaster alias definition. (See [alias_entry](#) for a discussion of minimal such postmaster alias definition in a modern, Unified Configuration setup.)

In older versions of the MTA, an alias was normally simply a valid [RFC 822 "local-part"](#); however, in more modern MTA configurations, with the [alias_domains](#) MTA option set to a value of 6, an alias consists of an entire address, including the domain name, rather than just the local-part. In particular, aliases must follow [RFC 822](#) syntax rules for local-parts (or addresses, when [alias_domains](#) has selected use of addresses); this means that for proper functioning, with the exception of periods which are specifically allowed in local-parts without quoting, the presence of any other [RFC 822 "specials"](#) character or a space in an alias will require that the alias be enclosed in double quotes, *e.g.*,

```
"John Doe": doe@acme.com
john.doe: doe@acme.com
```

Comment lines are allowed in the alias file. A comment line is any line that begins with an exclamation point, `!`, in column one.

Duplicate aliases (identical left hand sides) are not allowed in the alias file.

Note that prior to JES MS 6.1, certain sorts of errors in the format of aliases would not result in an immediate error message, but rather mail to the bad addresses would just be silently dropped. For instance, use of an apparently local (and syntactically unexceptional) but in fact non-existent user address as a value (on the right hand side) would not necessarily result in any error message---not if there was at least one apparently valid value on the right hand side. (This is in contrast to overt syntactic errors in the alias file format itself, which have always been reported at MTA process startup time, or at `imsimta cnbuild` time if a compiled configuration is in use. It is also in contrast to delivery problems to addresses that, at alias expansion time, appear potentially valid; delivery problems are and always have been reported back to the appropriate notification address, if any. It is also in contrast to the case where *all* of the values appear to be invalid.) As of JES MS 6.1, errors apparent at alias expansion time in aliases are reported to the other members of the alias. But in any case, when defining an alias it is a good idea to use `imsimta test -rewrite -check_expansions` to check aliases, and see [Alias restrictions](#) for further general information on alias operation and the alias file.

35.5.1.1 Alias file include files

Other files can be included in the primary alias file. A line of the form

```
<file-spec
```

directs the MTA to read the file `file-spec`. The file specification must be a complete file path specification and the file must have the same protections as the primary alias file; *i.e.*, it must be world readable.

The contents of the included file are inserted into the alias 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 alias file itself. Indeed, include files may themselves include other files. Up to three levels of include file nesting are allowed.

If a compiled configuration is being used, then the configuration must be [recompiled](#) and reinstalled before changes to any included file (or the primary alias file itself) will take effect. Note that this is not the case for mailing list membership files described in [Alias file mailing list aliases](#).

35.5.1.2 Alias file named parameters

This discussion describes alias file named parameters as set in legacy configuration in the [aliases file](#). In Unified Configuration, the equivalent settings are [alias options](#).

The `named-parameters` appearing in an alias file mailing list definition such as

```
alias: <file-spec, named-parameters, error-return-address, \  
      reply-to-address, errors-to-address, \  
      warnings-to-address, comments
```

or

```
alias: <ldap-url, named-parameters, error-return-address, \  
      reply-to-address, errors-to-address, \  
      warnings-to-address, comments
```

warnings-to-address, comments

or in an individual alias definition (see [Alias file format](#)) such as

alias: named-parameters, address-1, ..., address-n

are used to specify optional modifiers to the list expansion process. There can be zero or more named parameters, separated by commas, and they must appear before any positional parameters (*e.g., error-return-address, reply-to-address, etc.*). The general syntax of a named parameter is:

`[name] value`

Here `name` is the name of the parameter and `value` is its corresponding value. The square brackets are a mandatory part of the syntax: they do not indicate an optional field.

See [Alias header addition modifiers](#) for a description of controls on the effect of named parameters relating to the addition of headers, such as specifying whether a header is to be added only if not originally present, or added unconditionally, and whether the header supplements or substitutes for an originally present header.

The available named parameters are:

35.5.1.2.1 AND, OR

AND and OR control whether subsequent access control clauses (*e.g., [AUTH_LIST], [AUTH_MAPPING], etc.*) are ANDed or ORed. The default is controlled by the `or_clauses` MTA option--and is AND (for backwards compatibility) by default. For groups and lists defined in LDAP, see also the AND and OR values for the `mgrpBroadcasterPolicy` attribute (or more precisely, the attribute named by the `ldap_auth_policy` MTA option). For a more detailed discussion, see [Mailing list multiple access control interpretation](#).

In Unified Configuration, see the `alias_and` and `alias_or` alias options.

35.5.1.2.2 AUTH_CHANNEL, CANT_CHANNEL

AUTH_CHANNEL is used to specify a source channel or channels that may submit messages to the mailing list. CANT_CHANNEL is used to specify a source channel or channels that may not submit messages to the mailing list. The argument should be a (possibly wildcarded) channel name, or a space-separated list of (possibly wildcarded) channel names.

In Unified Configuration, see the `alias_auth_channel` and `alias_cant_channel` alias options.

35.5.1.2.3 AUTH_LIST, CANT_LIST, USERNAME_AUTH_LIST, USERNAME_CANT_LIST

AUTH_LIST is used to specify a list of addresses that are allowed to post to the mailing list. The `value` item must be either the full file path specification for a world readable file containing the list of addresses allowed to post to the list, or an [LDAP URL](#) that returns the list of addresses allowed to post to the list. The MTA will match the envelope From address against the addresses in the list; if no match occurs, the attempted posting fails and an error is returned to the would be postings originator. USERNAME_AUTH_LIST is analogous to AUTH_LIST, but for (possibly wildcarded) usernames rather than addresses; note that

usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting MTA process is running. Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, *, character. For instance, to specify that only the user JDOE may submit to a list, whether submitting from the L channel or via SMTP (e.g., from a POP or IMAP client that performs SASL SMTP authentication), the USERNAME_AUTH_LIST file would need to contain the entries:

```
JDOE
$*JDOE
```

where the first entry would match for messages submitted from the L channel and the second entry would match for messages submitted via SMTP AUTH. Note that as asterisk is normally a wildcard character, matching of only the exact literal asterisk character is specified by using the dollar character to quote the asterisk.

CANT_LIST has the opposite effect as AUTH_LIST: it supplies the full file path specification of a world readable file containing a list of addresses, or an LDAP URL returning a list of addresses, specifying which addresses may not post to the list. USERNAME_CANT_LIST is analogous to CANT_LIST, but for (possibly wildcarded) usernames rather than addresses; note that usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting MTA process is running.

One common use of this facility is to restrict a list so that only list members can post. This can be done by specifying the same file as both the list file and the AUTH_LIST file. For example, assuming that the list is named test-list and the list file is IMTA_TABLE:test-list.dis, the alias file entry would be:

```
test-list: <IMTA_TABLE:test-list.dis, \
          [auth_list] IMTA_TABLE:test-list.dis
```

For groups and lists defined in LDAP, the closest analogues are the `mgrpAllowedBroadcaster` and `mgrpDisallowedBroadcaster` attributes (more precisely, the attributes named by the `ldap_auth_url` and `ldap_cant_url` MTA options); and if wishing to compare against authenticated submission addresses, see also the `SMTP_AUTH_REQUIRED` value for the `mgrpBroadcasterPolicy` attribute (or whatever attribute is named by the `ldap_auth_policy` MTA option).

In Unified Configuration, see the `alias_auth_list`, `alias_cant_list`, `alias_username_auth_list`, and `alias_username_cant_list` alias options.

35.5.1.2.4 AUTH_MAPPING, CANT_MAPPING

AUTH_MAPPING and CANT_MAPPING are similar to AUTH_LIST and CANT_LIST except that they use mappings rather than explicit files of addresses. The `value` item associated with these named parameters is the name of a mapping table to use; the mapping is given the envelope From address as input.

If AUTH_MAPPING is used at least one mapping entry must match or the posting is rejected. If an entry does match the resulting string is checked; if it begins with an F, f, N, or n the

posting is rejected. The mailing list will expand normally if the resulting string begins with any other character.

If CANT_MAPPING is used, the posting is accepted if no entry matches. If an entry does match the resulting string is checked; if it begins with a T, t, Y, or y the posting is accepted. The posting is rejected if the resulting string begins with any other character.

The most common use of AUTH_MAPPING is to restrict postings to all users of a given (usually local) host. For example, if the local host name is ymir.claremont.edu, the following mailing list definition could be used for the gripes-list:

```
gripes: <pmdf_table:gripes-list.dis, [auth_mapping] x-gripes
```

The corresponding mapping file entries would be:

```
X-GRIPES
```

```
    *@ymir.claremont.edu    Y
```

Using a mapping table name beginning X- is recommended, so that this private mapping table name will not collide with a standard Oracle mapping table name.

In Unified Configuration, see the [alias_auth_mapping](#) and [alias_cant_mapping](#) alias options.

35.5.1.2.5 AUTH_USERNAME, CANT_USERNAME

AUTH_USERNAME is used to specify a username or wildcarded username pattern for an account or accounts allowed to post to the list. Note that this is generally only useful for senders submitting from the L channel or for senders who used the SMTP AUTH extension during their message submission; for messages submitted from other sources, the messages are considered to be submitted under the username of the MTA process that received and enqueued the message, *e.g.*, the account under which the MTA's SMTP server is running. Attempted postings from any other sender will be rejected.

CANT_USERNAME may be used to specify a username or wildcarded username pattern for an account or accounts whose postings should be rejected.

Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, *, character. Also note that the asterisk character is normally a wildcard, and must be quoted with the dollar character in order to be interpreted as a literal asterisk character. For instance, to specify that the only sender who may post to a list is user JDOE who will be submitted solely via SMTP with SMTP AUTH, you would use:

```
[AUTH_USERNAME] $*JDOE
```

Without the dollar sign, specifying just *JDOE would allow postings not only from user JDOE but also from any users AJDOE, BOBJDOE, *etc.*

For specifying more than one username (or wildcarded username pattern), see the USERNAME_AUTH_LIST and USERNAME_CANT_LIST parameters described above. For groups and lists defined in LDAP, the closest analogues are the `mgrpAllowedBroadcaster`

and `mgrpDisallowedBroadcaster` attributes (or more precisely, the attributes named by the `ldap_auth_url` and `ldap_cant_url` MTA options).

In Unified Configuration, see the `alias_auth_username` and `alias_cant_username` alias options.

35.5.1.2.6 BLOCKLIMIT, LINELIMIT

The `BLOCKLIMIT` and `LINELIMIT` parameters may be used to limit the size of messages that may be posted to the list. The `value` item must be an integer number of blocks for [`BLOCKLIMIT`], or an integer number of lines for [`LINELIMIT`]. The number of bytes in a block is specified via the `block_size` MTA option. The default value is 0, meaning that no limit is imposed on the size of message that may be posted to the list (apart, that is, from any channel or system wide limits). For user, groups, and lists defined in LDAP, see `mailMsgMaxBlocks` attribute (or more precisely, the attribute named by the `ldap_blocklimit` MTA option).

In Unified Configuration, see also the `alias_blocklimit` and `alias_linelimit` alias options.

35.5.1.2.7 CAPTURE, JOURNAL

(`CAPTURE` is new in JES MS 6.2; `JOURNAL` is new in Messaging Server 7.2-0.01.) The `CAPTURE` named parameter may be used to set an address to which to direct an encapsulated, "captured" copy of each message posted to the list. The `JOURNAL` named parameter works similarly, but generates an envelope "journal" format message. The `value` item should be the address to which to send the "captured" message copies. These parameters are exactly analogous to use of the LDAP attribute named by the `ldap_capture` MTA option on a group or mailing list defined via an LDAP entry.

In Unified Configuration, see also the `alias_capture` and `alias_journal` alias options.

35.5.1.2.8 CONVERSION_TAG

The `CONVERSION_TAG` named parameter may be used to set a `tag` which conversion file entries can match upon. The `value` item should be the string to use as the tag. For instance, if a list is defined

```
listname: </pmdf/table/listname.dis, [CONVERSION_TAG] listtag
```

then conversion file entries could include a `tag=listtag;` clause to match. For instance, if for some mailing list it was desired to convert any text/html parts in posted messages to text/plain, and if a site had an HTML to TEXT convertor called `htmltotextconvert` and had set up the conversion channel and a [CONVERSIONS mapping table](#) to apply to list postings, then a conversion file entry could be

```
in-chan=*; out-chan=*; in-type=text; in-subtype=html; tag=listtag;
out-type=text; out-subtype=plain; parameter-copy-0=*;
command="IMTA_PROGRAM:htmltotextconvert $INPUT_FILE $OUTPUT_FILE"
```

For users, groups, and lists defined in LDAP, see the `mailConversionTag` attribute (or more precisely, the attribute named by the `ldap_conversion_tag` MTA option).

In Unified Configuration, see also the `alias_conversion_tag` alias option.

35.5.1.2.9 CREATION_DATE

New in the 8.0 release.

The CREATION_DATE named parameter may be used to set a creation date for the alias (intended to be used for RRVs purposes). The creation date value must be in [RFC 3339 \(Date and Time on the Internet: Timestamps\)](#) format (a profile of [ISO 8601 format](#)), along the lines of:

```
YYYY-MM-DDTHH:MM:SS.ssZ
```

or

```
YYYY-MM-DDTHH:MM:SS.ssplus-or-minusHH:MM
```

where the hundredths of seconds portion is optional, and T and (if used) Z are not case sensitive. For instance:

```
2014-02-28T12:13:14.30-07:00
```

In Unified Configuration, see the [alias_creation_date](#) alias option. Or for users defined in LDAP, the analogous setting is controlled by whatever attribute is named by the [ldap_creation_date](#) MTA option, or at a domain level by whatever attribute is named by the [ldap_domain_attr_creation_date](#) MTA option.

35.5.1.2.10 DEFERRED, DEFERRED_LIST, DEFERRED_MAPPING

In Unified Configuration, see the [alias_deferred](#), [alias_deferred_list](#), and [alias_deferred_mapping](#) alias options.

The DEFERRED named parameter may be used to add a Deferred-delivery: header line. The value should be a date and time, in [ISO 8601 P format](#). Note that by default the MTA does not honor Deferred-delivery: headers; see the [deferred channel option](#) for a discussion.

The DEFERRED_LIST named parameter takes two (space-separated) values, a file specification for a list of originator addresses (or alternatively, a [URL](#) returning a list of addresses) to whose postings to add a Deferred-delivery: header, and the deferral date/time in [ISO 8601 format](#).

As of the 8.0 release (in prior versions, this feature "existed" but was not working), the DEFERRED_MAPPING named parameter may be used to run originator addresses through the specified mapping. DEFERRED_MAPPING takes one or two arguments, with a space between if the optional second argument is included. The first argument is required and must contain at a minimum the name of an MTA mapping table; the first argument may also, optionally, include a vertical bar character followed by a string to use as a prefix in the mapping table probe, prior to the originator address. The second argument is optional, consisting of a deferral date/time in [ISO 8601 format](#).

```
mapping-name[|probe-prefix] ISO-8601-deferral-time
```

Originator addresses will be run through the specified mapping. If the mapping template does not begin with an N, n, F, or f, and if it contains a valid date/time specification in [ISO 8601 format](#), then that date/time will be used as a deferral time. The default, if no mapping entry matches, or if an entry that begins with an N, n, F, or f, is not to add a Deferred-delivery: header. Note that the intended purpose of a `probe-prefix` is for convenience in using a single MTA mapping table for multiple mailing list deferral settings, *e.g.*, by using a probe prefix consisting of the list name, so that entries in the mapping table may be list specific.

Similarly, a deferral time specified as the second argument permits a default deferral time, that may then be overridden in the case of specific originators in the mapping table result.

Setting bit 3/value 8 of the `include_connectioninfo` MTA option will cause additional information to be included in the DEFERRED_MAPPING input probe. Thus if a `probe-prefix` has also been specified, then the probe will take the form:

```
transport-info|application-info|probe-prefix|originator-address
```

Note that by default the MTA does not honor Deferred-delivery: headers; see the [deferreddestination channel option](#) for a discussion. As a functionally preferable alternative to the Deferred-delivery: header line approach for retaining/deferring messages, see also the [SMTP SUBMIT FUTURERELEASE extension](#).

35.5.1.2.11 DELAY_NOTIFICATIONS, NODELAY_NOTIFICATIONS

The DELAY_NOTIFICATIONS named parameter requests that NOTARY delay notifications be sent for mailing list postings; the NODELAY_NOTIFICATIONS named parameter requests that NOTARY delay notifications not be sent for mailing list postings. The value specification is currently ignored and should always be NONE.

In Unified Configuration, see the [alias_delay_notifications](#) alias option. Or for users defined in LDAP, the analogous settings are controlled by whatever attribute is named by the [ldap_delay_notifications](#) MTA option, by default `mgrpDelayNotifications`.

35.5.1.2.12 DIGEST_RECURRENCE

RESTRICTED: Not yet fully implemented.

The DIGEST_RECURRENCE parameter takes an [ISO 8601](#) argument.

35.5.1.2.13 DIRECT_LIST, DIRECT_MAPPING

RESTRICTED: Not yet fully implemented.

35.5.1.2.14 ENVELOPE_FROM

This ENVELOPE_FROM parameter takes a required value specifying an address to replace the message's original envelope From address. This sets only the envelope From address, unlike the `error-return-address` positional parameter which also sets an Errors-to: address.

Setting the value to an address of the form `user+*@domain` has a special meaning. The asterisk character will be expanded into a representation of the recipient address; thus a separate copy of the list message is generated for each recipient, with each copy including the intended recipient address as a subaddress within the return address. If delivery errors subsequently occur, the subaddress will indicate which was the failing address. In some cases, when dealing with remote MTAs that generate nonstandard, uninformative delivery error messages, this can in theory be useful as a way of determining which recipient address(es) failed, even when the bounce message's inner content is relatively uninformative. And it may make processing of such bounce messages by an automated program more convenient. However, the tradeoff is that such per-user-specific return address values require that a separate message copy be generated and sent for each recipient; for a "large" list, with many recipients in the same destination domains, this can be a large increase in overhead (a large decrease in efficiency). And with more prevalent use nowadays of standard format notification messages, the "need" for this sort of approach, with its extra (potentially large) overhead, is

much less (since the intended recipient information can instead be extracted from the standard field in the contents of a standard format notification message).

(New in JES MS 6.3.) Setting the value to the forward slash character, /, has a special meaning. It tells the MTA to revert to using the original envelope From address that had been present on the incoming message, yet in all other respects use mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender.

In Unified Configuration, see the [alias_envelope_from](#) alias option. Or for groups and lists defined in LDAP, see the `mgrpErrorsTo` attribute (or more precisely, the attribute named by the `ldap_errors_to` MTA option).

35.5.1.2.15 ERROR_TEXT (string)

Specify a string to use as the "reason" which will be returned to the attempted sender if and when an attempted posting fails. For groups defined in LDAP, see the `mgrpRejectText` attribute or `mgrpMsgRejectText` attribute (or more precisely, whatever attribute(s) are named by the `ldap_reject_text` MTA option).

In Unified Configuration, see also the [alias_error_text](#) alias option.

35.5.1.2.16 EXPANDABLE, NONEXPANDABLE

The EXPANDABLE named parameter is used to specify that the associated list can be expanded (and hence its contents seen) by various protocols which may attempt such an operation. It does not mean, or imply, that the contents of the list will be expanded into message headers. The `value` specification is currently ignored and should always be NONE. The NONEXPANDABLE named parameter specifies that the associated list may not be expanded. Again, the `value` specified is currently ignored and should always be NONE. EXPANDABLE is the default, unless the `expandable_default` MTA option has been set, in which case the default is NONEXPANDABLE.

NONEXPANDABLE is useful in blocking the expansion of mailing lists via SMTP's EXPN command. Note that mailing list access controls, *e.g.*, AUTH_LIST, AUTH_MAPPING, *etc.*, also affect the expansion of mailing lists via SMTP's EXPN command; the SMTP server will only permit the EXPN if the SMTP client passes the access control (*e.g.*, has issued a prior MAIL FROM: command that passes the access control).

In Unified Configuration, see also the [alias_expandable](#) and [alias_nonexpandable](#) alias options.

35.5.1.2.17 EXPIRY

The EXPIRY named parameter is used to add an Expiry-date: header line. The value should be a date and time, in [ISO 8601 P format](#) (as described for the DEFERRED parameter above). (The MTA will convert the specified value into the appropriate corresponding [RFC 2822](#) date value needed for the header line.) The MTA's periodic return job will return messages whose Expiry-date: has passed.

For groups or lists defined in LDAP, see the `ldap_add_header` MTA option. In Unified Configuration, see also the [alias_expiry](#) alias option.

35.5.1.2.18 FILTER

The FILTER parameter takes a URL argument specifying the location of a Sieve filter to apply on attempted message postings. The argument may be any [supported form of URL](#) that makes

sense; in particular, besides supporting `file:file-spec` URLs or simply file specifications without the leading `file:`, LDAP URLs, and `data:sieve-commands` are also supported. Note that when specifying a file, it must be the full file specification for the filter file to apply.

In Unified Configuration, see the [alias_filter](#) alias option. Or for users defined in LDAP, the analogous setting is controlled by whatever attribute is named by the [ldap_filter](#) MTA option, by default `mailSieveRuleSource`.

35.5.1.2.19 HEADER_ADDITION, HEADER_TRIM

HEADER_TRIM may be used to add headers to or remove headers from posted messages. The argument must be a full file specification for a header trimming option file; see [Header option files](#) for information on the format of these files. HEADER_ADDITION is more specialized than HEADER_TRIM, being used when there are merely headers to be added. HEADER_ADDITION may be used to specify a file of headers to be added to posted messages. The argument must be a full file specification for the file containing headers to be added.

In particular, this facility can be used to add the standard mailing list headers defined in [RFC 2369](#). For instance, a site `domain.com` that has set up a list named `listname`, using the MAILSERV channel to manage subscription and unsubscription requests, and with certain list information and archives available at an FTP site, might use a header addition file along the lines of the following:

```
List-Help: <ftp://ftp.domain.com/pub/listname-help.txt> (FTP),
  <mailto:mailserv@domain.com?body=send%20/pub/listname-help.txt>,
  <mailto:mailserv@domain.com?body=help> (MAILSERV Instructions),
  <mailto:listname-owner@domain.com?subject=help> (List Manager)
List-Subscribe:
  <mailto:mailserv@domain.com?body=subscribe%20listname>
List-Unsubscribe:
  <mailto:mailserv@domain.com?body=unsubscribe%20listname>
List-Post: <mailto:listname@domain.com>
List-Owner: <mailto:listname-owner@domain.com?Subject=listname>
List-Archive: <ftp://ftp.domain.com/pub/listname/archive/>,
  <mailto:mailserv@domain.com?body=send%20/pub/listname/archive/*>
```

In Unified Configuration, see the [alias_header_addition](#) and [alias_header_trim](#) MTA options. Or for mailing lists defined in LDAP, see the LDAP attributes `mgrpAddHeader` and `mgrpRemoveHeader`, or more precisely, the LDAP attributes named by the [ldap_add_header](#) and [ldap_remove_header](#) MTA options.

35.5.1.2.20 HEADER_ALIAS, HEADER_EXPANSION

The HEADER_ALIAS named parameter forces the use of the original alias in any original headers constructed using this alias. HEADER_EXPANSION forces the alias to expand into its component addresses in any constructed header lines. The `value` specification is currently ignored and should always be `NONE`. These named parameters correspond to the `expand` and `no-expand` options for entries in personal alias databases. HEADER_ALIAS is the default for entries in the system alias file and database. Note that these parameters are only valid when headers are originally being constructed, as for instance for messages submitted via the L channel. These parameters are not relevant for incoming messages (such as incoming SMTP messages) for which the headers are already present in one form or another.

In Unified Configuration, see the [alias_header_alias](#) and [alias_header_expansion](#) alias options.

35.5.1.2.21 HEADER_CHECK

(New in 7.0.5) Used in conjunction with a `addrtypescan*` channel keyword. Valid arguments are `jettison` or `discard`. In Unified Configuration, see the [alias_header_check alias option](#). This named parameter is also analogous to the LDAP attribute named by the `ldap_check_header` MTA option.

35.5.1.2.22 HOLD_LIST, NOHOLD_LIST, HOLD_MAPPING, NOHOLD_MAPPING

The `HOLD_LIST` named parameter may be used to specify a list of originator addresses whose attempts to post to the list should be sidelined as `.HELD` messages. The `NOHOLD_LIST` named parameter may be used to specify the list of originator addresses whose postings should not be so sidelined, while all other postings will be sidelined. The value must be a [full file specification for a file of addresses, or an LDAP URL](#) returning a list of addresses. The `HOLD_MAPPING` and `NOHOLD_MAPPING` named parameters are used analogously, but via mapping tables rather than via lists. The value should be the name of an MTA mapping table.

In Unified Configuration, see the [alias_hold_list](#) and [alias_nohold_list](#) alias options.

35.5.1.2.23 IMPORTANCE, PRECEDENCE, PRIORITY, SENSITIVITY

The `IMPORTANCE`, `PRECEDENCE`, `PRIORITY`, and `SENSITIVITY` named parameters are used to generate respective headers; the value specification is inserted on the respective header line. In Unified Configuration, see the [alias_importance](#), [alias_precedence](#), [alias_priority](#), and [alias_sensitivity](#) alias options.

Note that the more general `HEADER_ADDITION` -- in Unified Configuration, the [alias_header_addition alias option](#) -- provides an alternate way to add these and other header lines. Or for aliases defined in LDAP, see the [ldap_add_header MTA option](#).

35.5.1.2.24 KEEP_DELIVERY, KEEP_READ

By default, the MTA strips delivery receipt and read receipt requests from messages posted to mailing lists. The `KEEP_DELIVERY` and `KEEP_READ` named parameters may be used to override this behavior, causing the MTA to retain any delivery receipt or read receipt requests, respectively, on messages posted to the list. The value specification is currently ignored and should always be `NONE`. Note that passing receipt requests through to mailing lists is quite dangerous; the default behavior of stripping such requests is *strongly* recommended.

In Unified Configuration, see the [alias_keep_delivery](#) and [alias_keep_read](#) alias options.

35.5.1.2.25 LIST_NAME

New in Messaging Server 7.4-0.01; RESTRICTED.

35.5.1.2.26 MODERATOR_ADDRESS, MODERATOR_LIST, MODERATOR_MAPPING, USERNAME_MODERATOR_LIST

The `MODERATOR_*` named parameters are used to establish a [moderated mailing list](#). All postings to the list not originating from a moderator are sent to the list's moderator. The address of the moderator must be specified with the `MODERATOR_ADDRESS` named

parameter. The moderator address determines where moderator mail is sent when someone other than the moderator posts. The value of that named parameter is the moderator's address. For example,

```
test-list: <IMTA_TABLE:test.dis, \  
          [MODERATOR_ADDRESS] bob@domain.com
```

When there may be multiple moderator addresses (for instance, both robert@mail1.domain.com and bob@domain.com), use MODERATOR_LIST, USERNAME_MODERATOR_LIST, or MODERATOR_MAPPING to specify all addresses from which postings should be passed directly to the list and not sent to the list's moderator. MODERATOR_LIST specifies either the name of a file containing a list of moderator addresses, or an LDAP URL returning a list of moderator addresses. USERNAME_MODERATOR_LIST specifies either the name of a file containing a list of (possibly wildcarded) moderator usernames, or an LDAP URL returning a list of (possibly wildcarded) moderator usernames; note that usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting MTA process is running. Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, *, character. For instance, to specify that only the user JDOE is the list moderator, whether submitting from the L channel or via SMTP (*e.g.*, from a POP or IMAP client that performs SASL SMTP authentication), the USERNAME_MODERATOR_LIST file would need to contain the entries:

```
JDOE  
$*JDOE
```

where the first entry would match for messages submitted from the L channel and the second entry would match for messages submitted via SMTP AUTH. Note that as asterisk is normally a wildcard character, matching of only the exact literal asterisk character is specified by using the dollar character to quote the asterisk.

MODERATOR_MAPPING specifies the name of a mapping table used to verify whether or not an address is a moderator address.

If a MODERATOR_LIST or MODERATOR_MAPPING parameter is used, thereby specifying who may post directly to the list, then a MODERATOR_ADDRESS parameter should also be present to specify the address to which to send postings not from any moderator.

The use of the MODERATOR_ADDRESS parameter alone, without the MODERATOR_LIST parameter, is equivalent to using MODERATOR_ADDRESS and a MODERATOR_LIST consisting of just the one moderator address.

Unified Configuration has analogous alias options [alias_moderator_address](#), [alias_moderator_list](#), [alias_moderator_mapping](#), and [alias_username_moderator_list](#). Or for lists defined in LDAP, see the `mgrpMsgRejectAction` and `mgrpModerator` attributes, or more precisely whatever LDAP attributes are named by the [ldap_reject_action](#) and [ldap_moderator_url](#) MTA options.

35.5.1.2.27 NOSOLICIT (comma-separated list of strings)

New in JES MS 6.2. Set a solicitation keyword, or a list of solicitation keywords, that will not be allowed on postings to the list. Attempted postings that have such a keyword set will be rejected with "Solicitation check failure on SOLICIT=*keyword*" error text.

In Unified Configuration, see the [alias_nosolicit](#) alias option. Or for lists defined in LDAP, see the [ldap_nosolicit](#) MTA option.

35.5.1.2.28 OPTIN, OPTIN1, OPTIN2, OPTIN3, OPTIN4, OPTIN5, OPTIN6, OPTIN7, OPTIN8

Set optin values for spam filtering.

In Unified Configuration, see the [alias_optin*](#) alias options.

For aliases/lists defined in LDAP, see the [ldap_optin*](#) MTA options.

35.5.1.2.29 ORIGINATOR_REPLY, NOORIGINATOR_REPLY

ORIGINATOR_REPLY is used to control whether or not the originator's address is added to any generated Reply-to: header. The `value` item should be the [full file path specification for a world readable file, or a resolvable URL](#), containing the list of addresses that should never be added. (This is usually the mailing list itself.) The MTA will match the envelope From address against the addresses in the list; if no match occurs, the originator's address will be added to any generated Reply-to: header.

NOORIGINATOR_REPLY specifies that any generated Reply-to: header should contain only explicitly specified addresses. The `value` item is ignored. NOORIGINATOR_REPLY is the default.

In Unified Configuration, see the [alias_originator_reply](#) and [alias_nooriginator_reply](#) alias options.

35.5.1.2.30 PASSWORD

Specify a password, or a comma-separated list of passwords, that allow posting to the list. An attempted posting to the list must contain one of these values on an Approved: header line in order for the posting to be allowed. During mailing list expansion, the password value will be removed from the Approved: header line; indeed, if that is the only value on the Approved: header line, then the entire header line will be removed. See [Password-protected mailing lists](#).

In Unified Configuration, see the [alias_password](#) alias option.

For aliases/lists defined in LDAP, see the [ldap_auth_password](#) MTA option.

35.5.1.2.31 PREFIX_TEXT, SUFFIX_TEXT

(New in JES MS 6.0.) Insert prefix or suffix text into messages as they undergo list expansion. Prior to Messaging Server 7.0 update 3, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.0 update 3, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The attribute values are given in UTF-8; this is then converted to match the charset of the part into which the text is being inserted.

In Unified Configuration, see the [alias_prefix_text](#) and [alias_suffix_text](#) alias options. Or for lists defined in LDAP, see the `mgrpMsgPrefixText` and `mgrpMsgSuffixText` attributes, or more precisely whatever attributes are named by the

[ldap_prefix_text](#) and [ldap_suffix_text](#) MTA options. More generally, for adding prefix or suffix text to *any* message, not just postings to groups or lists, see the [Sieve addprefix and addsuffix extensions](#).

35.5.1.2.32 PUBLIC, PRIVATE

The PUBLIC named parameter specifies that the associated alias is public and hence can appear in any constructed header lines. The `value` specification is currently ignored and should always be NONE. The PRIVATE named parameter specifies that the alias is private and should appear as an empty group construct in message headers. The `value` specification is used as the name for the group. Neither PUBLIC nor PRIVATE have any effect if the HEADER_EXPANSION named parameter is also specified. These named parameters correspond to the public and private options for entries in personal alias databases. PUBLIC is the default for entries in the system alias file and database.

Note that these parameters are only valid when headers are originally being constructed, as for instance for messages submitted via the L channel. These parameters are not relevant for incoming messages (such as incoming SMTP messages) for which the headers are already present in one form or another.

In Unified Configuration, see the [alias_private](#) and [alias_public](#) alias options.

35.5.1.2.33 RECEIVEDFOR, NORECEIVEDFOR, RECEIVEDFROM, NORECEIVEDFROM

These named parameters control features of what appears in the Received: header constructed when expanding the alias, and override normal channel [receivedfor](#), [noreceivedfor](#), [receivedfrom](#), or [noreceivedfrom](#) channel option settings. The `value` specification is currently ignored and should always be NONE.

In Unified Configuration, see the [alias_receivedfor](#), [alias_noreceivedfor](#), [alias_receivedfrom](#), and [alias_noreceivedfrom](#) alias options.

35.5.1.2.34 REPROCESS

The REPROCESS named parameter is used to request deferred expansion of the mailing list, where rather than expanding the mailing list "on line", the message should instead be enqueued to the reprocess channel; the [reprocess channel](#) can then perform the mailing list processing in a separate step. The `value` specification is currently ignored and should always be `reprocess`.

Use of this parameter defers much of the processing overhead of handling the message to the later step when the [reprocess channel](#) runs, rather than doing the processing as the message is initially accepted. This deferred processing can be especially helpful in cases such as incoming SMTP messages addressed to large mailing lists, where "on line" delays could lead to connection time outs.

Use of this parameter as in:

```
listname: </pmdf/table/listname.dis, [REPROCESS] reprocess
```

thus provides essentially identical functionality as defining a mailing list in two stages through the reprocess channel to obtain deferred expansion (the mailing list addresses aren't even expanded until the reprocess channel runs) such as:

```
listname:          listname-expand@reprocess
listname-expand:  </pmdf/table/listname.dis
```

In Unified Configuration, the analogous alias option is [alias_reprocess](#). Or for aliases defined in LDAP, see the `mailDeferProcessing` attribute, or more precisely whatever LDAP attribute is named by the [ldap_reprocess](#) MTA option.

35.5.1.2.35 SASL_AUTH_LIST, SASL_AUTH_MAPPING, SASL_CANT_LIST, SASL_CANT_MAPPING, SASL_MODERATOR_LIST, SASL_MODERATOR_MAPPING

These named parameters are analogues of the non-SASL named parameters, but with the additional requirement that an authenticated address be present in the message (whether that be a address literally authenticated via SMTP AUTH, or forced via, *e.g.*, an [authrewrite](#) or [FROM_ACCESS](#) effect).

In Unified Configuration, see the [alias_sasl_*](#) alias options.

35.5.1.2.36 SEQUENCE_PREFIX, SEQUENCE_SUFFIX, SEQUENCE_STRIP

The `SEQUENCE_PREFIX` and `SEQUENCE_SUFFIX` named parameters request that a sequence number be prepended or appended to the Subject: lines of messages posted to the list. The `value` item gives the full file path specification of a sequence number file. This file is read, incremented, and updated each time a message is posted to the list. The number read from the file is prepended, in the case of `SEQUENCE_PREFIX`, or appended, in the case of `SEQUENCE_SUFFIX`, to the message's Subject: header line. This mechanism provides a way of uniquely sequencing each message posted to a list so that recipients can more easily track postings and determine whether or not they have missed any.

By default, a response to a previously posted message (with a previous sequence number) retains the previous sequence number as well as adding a new sequence number to the subject line; the build up of sequence numbers shows the entire "thread" of the message in question. However, the `SEQUENCE_STRIP` named parameter can be used to request that only the highest numbered, *i.e.*, most recent, sequence number be retained on the subject line. The `value` item is currently ignored and should always be `NONE`.

Important note: To ensure that sequence numbers are only incremented for successful postings, a `SEQUENCE_PREFIX` or `SEQUENCE_SUFFIX` named parameter should always appear as the last named parameter; that is, if other named parameters are also being used, the `SEQUENCE_*` named parameter should appear at the end of the list of named parameters.

Sequence number files are binary files and must have the proper file attributes and access permissions in order to function correctly.

In Unified Configuration the analogous alias options are [alias_sequence_prefix](#), [alias_sequence_suffix](#), and [alias_sequence_strip](#).

35.5.1.2.37 SINGLE

Force a separate message copy per recipient (per list member). Thus it can be considered a per-list analogue of the [single](#) channel option.

In Unified Configuration, its analogue is the [alias_single](#) alias option.

35.5.1.2.38 SPARE1,....,SPARE18

(New in Messaging Server 7.0 update 2) Analogous to the attributes named by the [ldap_spare_N](#) MTA options. In Unified Configuration, the analogous alias options are [alias_spare*](#).

35.5.1.2.39 TAG

The TAG named parameter may be used to prefix specified text to the Subject: header of posted messages. The value item should be the string to be added. The string should not contain the vertical bar, |, character; prior to JES MS 6.3, the string should not have contained the space character. For instance,

```
schedule-list: <d1:[adam]schedule-list.dis, [TAG] Schedule posting -- , \  
[AUTH_LIST] d1:[adam]schedule-list.dis
```

will cause any postings to the list schedule-list to have a Subject: header that begins "Schedule posting -- " followed by whatever the original subject of the posting might have been. See the [ldap_add_tag](#) MTA option for setting an attribute name to provide analogous functionality for lists defined in LDAP, and in Unified Configuration, see also the [alias_tag](#) alias option.

35.5.1.2.40 TO

The TO named parameter specifies what to put on the To: header line of postings to the mailing list.

In Unified Configuration, see the [alias_to](#) alias option.

35.5.1.2.41 USERNAME

The USERNAME named parameter may be used to set the "username" that the MTA will consider to "own" these mailing list messages. The [imsimta qm utility](#) will allow that username to inspect and bounce messages in the queue resulting from expansion of this mailing list. The value item should be the username of the account to "own" the mailing list postings.

In Unified Configuration, see the [alias_username](#) alias option.

35.5.1.3 Alias file mailing list aliases

A mailing list address may be defined in the alias file or alias database by:

- Specifying a list of translation values for an alias, rather than simply a single translation value for the alias;
- Specifying an envelope From override address -- an [\[ENVELOPE_FROM\] named parameter](#). (If no such envelope From override address is specified, then technically an alias with multiple translation values corresponds to a mail group -- an auto-forwarder forwarding to multiple recipients -- rather than, strictly speaking, a mail list.)

A mailing list address [alias](#) with associated mailing list file [file-spec](#) or [LDAP URL](#) [ldap-url](#) is specified in the alias file with an entry of, respectively, the general form

```
alias: <file-spec, optional-parameters
```

or

```
alias: <ldap-url, optional-parameters
```

Similar definitions may also be made in the alias database, (though of course omitting the colon, as just white space separates the alias from its definition in the alias database).

Mailing lists have many options associated with them; for a full discussion of mailing list aliases, see [Mailing_lists](#), or for a discussion of the optional named parameter frequently used on mailing list alias definitions, see [Alias file named parameter](#).

35.5.1.3.1 Alias file LDAP URL alias values

An alias value (that is, the right hand side of an alias definition) may be specified either as an address directly, *e.g.*, `user@domain`, or indirectly referencing an LDAP URL---specifically, an LDAP search URL---that returns one or more addresses. The format is

```
alias: <ldap-url
```

Note that this is just a special case of use of an LDAP URL for a mailing list definition, as mentioned in [Alias file mailing list aliases](#): the LDAP query URL may be such as to return only one address rather than multiple addresses, and all of the optional mailing list parameters are omitted. Also note that if desiring to look up all incoming local channel addresses in an LDAP directory using some consistent addressing and URL format, it is generally simpler to configure such lookups globally using the `alias_urlN` options. However, the special case of looking up just a few individual local channel addresses in an LDAP directory via their own individual LDAP query URLs is of sufficient interest to warrant further discussion.

Standard LDAP URLs are used, typically with the host and port omitted; the host and port are instead typically specified with the `ldap_host` and `ldap_port` MTA options. (As of Messaging Server 7.0u4, the LDAP server host and port may instead be specified in the LDAP URL itself.) That is, the LDAP URL would typically 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 what information to return. For an alias, the desired `attributes` to specify returning would typically be the `mail` 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 any object that has the "objectclass=person" and "cn=John Smith" attribute-value pairs.

For instance, at a site `domain.com` with an LDAP server running on port 389 of the system `ldap.domain.com`, the MTA option file might have the lines

```
LDAP_HOST=ldap.domain.com
LDAP_PORT=389
```

set, and an alias file line might appear as:

```
John.Smith@domain.com: <ldap:///o=domain.com?mail?sub?(&(objectClass=person)(cn=John%20Smith))
```

The Unified Configuration equivalent would be:

```
msconfig> show ldap_host
role.mta.ldap_host = ldap.domain.com
msconfig> show ldap_port
role.mta.ldap_port = 389
msconfig> set alias:John\.\Smith@domain\.\com.alias_entry '<ldap:///o=domain.com?mail?sub?(&(objectClass=person)(cn=John%20Smith) '
msconfig> show alias:John\.\Smith@domain\.\com
role.alias:John\.\Smith@domain\.\com.alias_entry = <ldap:///o=domain.com?mail?sub?(&(objectClass=person)(cn=John%20Smith))
```

Note that certain characters, such as for instance space characters, should be encoded in URLs according to the URL character encoding rules of [RFC 1738](#).

35.6 Alias database

The MTA's alias database, seldom used nowadays, provided an additional location for storing large numbers of [aliases](#), supplementing the the [alias options](#) (Unified Configuration) or [alias file](#) (legacy configuration). Nowadays, with "Direct LDAP" configuration, the majority of aliases are normally [stored in LDAP](#).

The [use_alias_database](#) MTA option controls whether or not the MTA makes use of the alias database. The default is 1, so the mere presence of the alias database activates the MTA's use of it as a source of aliases.

35.6.1 Using another alias source and the alias database

The alias database is a *supplement* to the [alias options](#) (Unified Configuration) or [alias file](#) (legacy configuration); it is *not a replacement* for them. If the alias database exists, the MTA uses *both* the usual alias source (the alias options in Unified Configuration, or the alias file in legacy configuration) *and* the alias database.

The alias database is consulted once each time the alias options/regular alias file is consulted. However, the alias database is checked *before* the alias options/regular alias file is consulted. In effect, the database acts as a sort of address rewriter that is invoked prior to using the regular alias source. Although duplicate entries are allowed in the database, it is undefined as to which of the duplicate entries will be returned when the database is accessed. Database entries are case insensitive.

The fact that limited [recursion](#) is allowed in the [alias options/alias file](#) makes the complete translation mechanism rather complex. For example, suppose that the alias file contains the entries,

```
A: C, J
B: D, K
D: G, H
E: I
```

and the alias database contains the entries,

```
D: E
C: B
F: D
```

Now suppose the address `A@local-host` was presented to the MTA. First `A` would be looked up in the alias database --- not found. Then `A` would be translated into `C` and `J` by the alias file. `C` would in turn be translated into `B` by the alias database while `J` would remain unchanged. `B` would then be translated into `D` and `K` by the alias file. `D` would then be translated into `E` by the alias database while `K` would remain unchanged. Finally, `E` would be translated into `I` by the alias file, and since `I` does not appear in the alias database the process would terminate. The final result is that `A` translates into the list `I, J, K`.

The easiest way to look at the translation process is to simply follow it step-by-step as illustrated below.

Initial look up	Data base	Data File	Data base	Data File	Data base	Data File	Data base	Result
	A	C	B	D	E	I	I	I
				K	K	.	.	K
		J	J	J
B	B	D	E	I	I	.	.	I
		K	K	K
C	B	D	E	I	I	.	.	I
		K	K	K
D	E	I	I	I
E	E	I	I	I
F	D	G	G	G
		H	H	H

Such complex use of the aliases facility is not encouraged and is presented for illustrative purposes only.

Note: In particular, for most normal goals any particular entry should appear in either an [alias option](#)/the [alias file](#) or the alias database, *not in both!*

35.6.2 Alias database format

In early versions of the MTA, the format of the alias database was an on-disk database, built using the [imsimta crdb utility](#) based upon a flat text file input. Alternatively, new in the 8.0 release, the MTA supports use of memcache for certain database/storage uses, including the alias database; see the [alias_database_url](#) MTA option.

Indeed, the alias database can be considered to have the same format as the optional [domain database file](#). The allowed format of the flat text input file is normally:

key value

one entry per line, with the *key* beginning in column one, one or more white space (SP or TAB) characters, and then the *value* on the right hand side. The *key*, that is, the alias, is limited to 32 characters in length and can translate to a *value* string containing at most 80 characters unless either a "long" or a "huge" database is used. See the [-long_records](#) and [-huge_records](#) switches of the [imsimta crdb utility](#) for information on long databases, and on huge databases.

Length restrictions aside, alias database entries are handled in the same way as [alias file](#) entries and can be used in exactly the same way. Both multiple addresses and mailing list

references are allowed. (Note that in long or huge alias databases, while the translation string may contain 256 or 1024 characters, respectively, any individual address appearing in the translation string is limited to at most 256 characters (252 characters in iMS 5.2 and earlier). The purpose of the longer translation string limit in such databases is to allow room for multiple comma-separated addresses, or for mailing list definitions that besides an "address", also contain additional named or positional parameters.)

The `comment_chars` MTA option controls which characters (by default exclamation point and semicolon) in column one of a line are considered to indicate a comment line. The left angle character may be used to read another file into the alias database text input file.

The alias database, like the alias file, must be world readable.

The MTA alias database is created from an input text file (*not* from the `alias file`--from a *different* input text file) using the `imsimta crdb` utility. The format of entries in the input file for `crdb` should be:

```
alias1  alias-value1
alias2  alias-value2
.       .
.       .
.       .
```

Note that unlike the aliases file, the entries in the alias database source text file normally do not use a colon to separate the alias from its value.

On UNIX systems, use the commands

```
# imsimta crdb input-file-spec IMTA_ROOT:data/db/aliasesdb-tmp
# imsimta renamedb IMTA_ROOT:data/db/aliasesdb-tmp IMTA_ROOT:data/db/aliasesdb
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated. (Note that the "symbolic" name `IMTA_ROOT` can be used in such a command.)

Alternatively, a source file using colons, (that is, of the same format as the alias file), *e.g.*,

```
alias1:  alias-value1
alias2:  alias-value2
.       .
.       .
.       .
```

may be used providing that the `-strip_colons` switch is used when building the database; *e.g.*, on UNIX:

```
# imsimta crdb -strip_colons input-file-spec IMTA_ROOT:data/db/aliasesdb-tmp
# imsimta renamedb IMTA_ROOT:data/db/aliasesdb-tmp IMTA_ROOT:data/db/aliasesdb
```

35.7 Subaddresses in aliases

As background on the purpose of subaddresses, the MTA interprets a `+ character` in an address specially: in an address of the form `name+subaddress@domain` the MTA considers

the portion of the mailbox after the plus character a *subaddress*. If the MTA tells the [Message Store](#) to "trust" that subaddress as a folder name for delivery purposes (see in particular the [fileinto](#) channel option, and the [deliveryflags](#) channel option), then Message Store too will treat the subaddress specially, delivering straight to that folder.¹

When looking up an alias, the use of subaddresses introduces an extra factor. The MTA's "I" channel, or any channel marked with the [aliaslocal](#) channel option, will try looking up aliases.

Subaddresses in aliases are handled as follows. By default, (that is, with the [subaddressrelaxed](#) channel option explicitly or implicitly on the channel doing the alias lookup), the MTA first checks for an alias entry including the subaddress; if no such entry is found, the MTA next checks for an entry with an asterisk, *, in place of the subaddress. Finally, if there is no prior match, the MTA checks for an entry without any subaddress. For instance, alias entries

```
adam+privileged:    system
adam:               bob+*
carl+special:       system
carl+*:             david+*
carl:               eric
```

cause the MTA to translate adam+privileged to system, and adam to bob (note the special case handling whereby the MTA removes the trailing subaddress character, +, from the translation value bob+* if there is in fact no subaddress), while adam+talklist, adam+general, *etc.*, will be translated to bob+talklist, bob+general, *etc.* carl+special will be translated to system and carl to eric, while carl+talklist, carl+general, *etc.*, will be translated to david+talklist, david+general, *etc.*

This handling of subaddresses during alias lookups is configurable; see the [subaddress*](#) channel options for configuration at the channel level, or for aliases stored in LDAP see the (new in MS 8.0) domain-level control available via the [ldap_domain_attr_subaddress](#) MTA option.

¹ Note that the [ims-ms channel](#)'s support for folder delivery can be disabled via the [FILEINTO ims-ms-channel-specific option](#).

35.8 Alias special formats

In general, alias "special formats" supported for aliases stored in the [alias file](#) are also supported for [aliases stored via an alias group in Unified Configuration](#). But only some of the alias "special formats" supported for such aliases (those stored in the [alias file](#)) are also supported for [aliases stored in LDAP](#).

In particular, for all forms of alias:

- an alias whose value begins with a leading colon has a special interpretation in regards to generation of notification messages;
- [subaddress support](#) is available;
- the [aliaswild](#) effect (perform a catchall * alias probe if no exact alias is found) is available (though use for LDAP aliases is strongly discouraged as for the case of LDAP aliases, use

of the supported domain level attribute `mailDomainCatchallAddress` is recommended instead).

Furthermore, for aliases stored via an `alias` group in Unified Configuration:

- the `mail_off` MTA option feature is supported;
- the `post_off` MTA option feature is supported;

See the discussion of such special formats for [aliases in the alias file](#) for further details.

35.9 Alias header addition modifiers

The action of those [alias options](#) (Unified Configuration) or [alias file named parameters](#) (legacy configuration) that can add headers, *e.g.*, alias options such as `alias_deferred` or `alias_priority`, or similarly alias file named parameters `[DEFERRED]`, `[PRIORITY]`, *etc.*, can be modified by the special characters shown in [Table of alias header addition modifiers](#), by appending the special character at the end of the value for the option or parameter.

Table 35.2 Alias header addition modifiers

Character	Description
	Insert if not already present; inserts as a Resent- if already present
*	Only insert if not already present
&	Insert if not already present; add to old field if already present
^	Delete any old field present; always insert the new field
\	Delete old field and don't insert a new one

35.10 Alias recursion and nested list definitions

Aliases may reference other aliases, [in LDAP](#), in the [alias database](#), in the [alias file \(legacy configuration\)](#), and in [alias named group Unified Configuration option](#) settings. To avoid possible infinite recursion reference loops, the MTA limits such nested or recursive references to a default maximum of ten levels (see the `max_alias_levels` MTA option).

If an alias references itself, either directly or indirectly, an alias loop results. The loop eventually terminates due to the level restriction, but the termination conditions may not produce consistent results in all cases.

The special case of an alias directly referencing itself is allowed and specially handled. For example, the [alias file](#) definition

```
alias-name: alias-name, other-address-1, other-address-2, ...
```

will expand `alias-name` into itself plus `other-address-1`, `other-address-2`, and so on. `alias-name` may in turn get expanded in some other way (the system [alias database](#) or personal alias database) but it will not be expanded further by the alias file.

Note that [implicit domain name use](#) (having the MTA itself insert its default domain name onto "bare" usernames) may affect the "matching" of alias names needed for the MTA's special code to trigger. In order for a "match" to be assured, either use a "bare" username on both the left and right hand sides, or use a fully-qualified address on both the left and right hand sides.

35.11 Alias restrictions

There are some important restrictions that should be observed when using aliases, especially aliases in the alias file or alias database.

35.11.1 General alias restrictions

1. The addresses in the [alias file](#) or [alias database](#) or [stored in LDAP attributes such as mail, mailAlternateAddress, or mailEquivalentAddress](#) should be formatted as pure [RFC 822](#) addresses, *e.g.*, `user@domain-name`. Do not try to use DECnet or other routing conventions that you can get away with in the rewrite rules table. Not only may such things fail, they may not produce a visible error (see the next item). Source routes are the only exotica that are permitted.
2. Certain types of bogus addresses in a group or list alias would not, prior to JES MS 6.1, generate a "bad address" return message. Specifically, if, for a given address in the group or list, the system name was illegal or there was a syntax error in the address specification, then the copy of the message to that address might be silently dropped and no one will be the wiser. In the case of mailing lists defined in the alias file or alias database, if the mailing list membership file associated with an alias does not exist, then mail to the list itself may be dropped. However, errors in the mailbox part of the address (*e.g.*, "no such user") would be handled correctly. As of JES MS 6.1, there is enhanced handling for such cases. Errors in e-mail addresses in group definitions (but not errors in LDAP DN or LDAP URLs, when group members are referenced via LDAP DN or LDAP URL), but where at least one apparently valid address is in the group, will be reported to (the rest of) the group; in the list case, the error will be reported to the list's notification address. Note that errors in LDAP DN or LDAP URL, when group members are defined/referenced via such, will cause expansion of the group to abort. However, as of 7.0.5, errors in LDAP DN or LDAP URL during group access checking (during expansion of the group allowed to post to the group) will be ignored (and processing of the group access check will continue); previously such errors halted the expansion process for the access group. System managers should take care to test each list they set up to insure that all the recipient addresses are correct. The `imta test -rewrite -check_expansions` utility provides a way to do such checking of syntactic correctness of list definitions and list membership addresses. Groups and lists should be checked periodically and also whenever extensive changes are made.
3. Aliases in the [alias file](#) can contain up to 60 characters. Aliases in the [database](#) can contain up to 32 characters in a short database, up to 80 characters in a long database, and up to 256 characters (252 characters in iMS 5.2 and earlier) in a huge database. In the alias file, the addresses to which aliases translate can contain up to 256 characters (252 characters in iMS 5.2 and earlier). In the case of a short database, the translation value can contain up to 80 characters; in the case of a long database the translation value can contain up to 256 characters; in the case of a huge database the translation value can contain up to 1024 characters. In some cases failing to observe length restrictions may lead to addresses being silently dropped from lists.
4. The LDAP URL template value to which a `alias_urlN` MTA is set is limited to 256 characters (252 characters in iMS 5.2 and earlier) before substitutions; the substitutions

may insert additional material and the length after such substitutions is limited to 1024 characters. Note that the substitution of "known" attributes when asterisk, *, is specified as the attribute-to-return is not considered as part of the regular substitution; this substitution is performed at a later step and the length after this "known" attributes substitution is limited to 4096 characters.

35.11.2 Additional LDAP alias restrictions

1. For performance reasons, the MTA normally [caches](#) the results of LDAP queries. Also, the LDAP server itself normally does some caching of searches. So changes to an [LDAP alias](#) will not always be "immediately" apparent to already running MTA processes.

35.11.3 Additional alias file (or database) restrictions

1. The MTA reads the [alias file](#) only as each program using the MTA initializes itself. This means that if you are using a permanently resident server (such as the SMTP server) you should be sure to stop and restart the server each time the alias file or any of the files it includes is changed -- first [recompiling](#) the MTA configuration, if you are using a compiled configuration (since a compiled configuration includes the alias file). (The `imsimta restart` utility provides a simple way to restart any such MTA detached processes.) On the other hand, mailing list membership files referenced by the alias file are read and reread as needed, so servers need not be restarted when one of these files is changed.
2. The alias file is always read into memory in its entirety each time the MTA is used. All files included by the primary alias file are also loaded into memory. (Mailing list membership files are not loaded into memory.) The use of a huge alias file can eat up lots of memory. Liberal use of the mailing list membership reference operator, `<`, to reference long lists is recommended. Long lists of addresses coded directly into the alias file or any files it includes should be avoided. Use of an alias database for large numbers of aliases is also recommended.

35.12 Address reversal

After address rewriting via the MTA's [rewrite rules](#), header From: addresses and other backwards-pointing addresses and forwards-pointing header addresses normally receive one additional processing step.¹ This additional processing step is referred to as *address reversal*. Another term used is *address canonicalization*, since address reversal is most commonly used to change possible alternate address forms into a single, canonical form. Address reversal can be performed via [LDAP lookups](#), and/or via use of a [reverse database](#) and/or [REVERSE mapping](#). Note that an LDAP lookup, if specified via the [reverse_url](#) MTA option, is performed prior to checking the reverse database and/or REVERSE mapping.

Special handling of [subaddresses](#) is available during address reversal. And special handling of what might be termed address "decorations", namely [RFC 822 comment strings and personal names](#), is also available.

New in 8.0, address reversal can be made sensitive to exactly which header field (*e.g.*, From: *vs.* Sender:, *etc.*) is being processed by setting a special bit in the [use_reverse_database](#) MTA option which will cause inclusion of the header field name in [REVERSE mapping table](#) probes.

The primary use of address reversal is to substitute a generic, standardized address for internal or host-specific addresses. Address reversal is a particularly powerful tool when used in conjunction with aliases.

¹ Address reversal processing *can* be restricted in various ways. Address reversal can be restricted to only backwards pointing addresses if bit 2/value 4 in the MTA option [use_reverse_database](#) is cleared. Application of address reversal processing to envelope From address can be disabled using the [reverse_envelope MTA option](#). The [noreverse_channel option](#) can disable address reversal from being performed during enqueues to particular destination channels. However, in modern MTA configurations using LDAP-based aliases, a great many functions are critically dependent upon the MTA performing its LDAP lookup address reversal; before considering any restrictions upon normal address reversal, consider carefully the discussion of [Intended side effects of LDAP address reversal](#).

35.12.1 LDAP lookups for address reversal

If the [reverse_url MTA option](#) has been set, then each address (other than envelope To addresses) passing through the MTA will be checked against the result of an LDAP query constructed as specified by the [reverse_url](#) option (querying the LDAP server at the port specified by the [ugldaphost](#) and [ugldappport](#) options, which may be overridden by the MTA-specific [ldap_host](#) and [ldap_port](#) MTA options). If the LDAP query succeeds and returns a value, that value will be substituted in place of the original address.

For the [reverse_url MTA option](#), standard LDAP URLs as per [RFC 2255](#) must be used, except with the host and port normally omitted, as the host and port are normally instead specified via the base or MTA-specific option settings mentioned above. That is, the LDAP URL is typically specified along the lines of:

```
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; it might correspond to the organization's top level in the Directory Information Tree, or it might correspond to a subset of the organization, based upon the domain name in the original address. The optional `attributes`, `scope`, and `filter` portions of the URL further refine what information to return. For address reversal, the desired `attributes` to specify returning would typically be the `mail` attribute (or some similar attribute). The `scope` may be any of `base` (the default), `one`, or `sub`. And the desired `filter` would typically be based upon the mailbox (local portion) of the incoming addresses.

Certain substitution sequences may be used to construct the LDAP search URL; see [Table of LDAP URL substitution sequences](#) in [LDAP URL substitution sequences](#) for details.

35.12.1.1 Intended side effects of LDAP address reversal

Doing a [reverse_url](#) lookup actually has effects beyond pure address reversal. (And this is why a [reverse_url](#) lookup normally uses the [\\$R substitution](#) for a filter that searches for a given address as the canonical `mail` attribute, as well as searching for the attributes that would actually require address reversal: one wants the [reverse_url](#) lookup to find an entry even for an address that is already in canonical form.) The recommended setting for the [reverse_url MTA option](#) makes use of the LDAP URL `$N` substitution to specify an extensive list of attributes to be fetched; so [reverse_url](#) lookups also normally make use of (or at least fetch and cache) the attributes named by the MTA options:

- [ldap_primary_address](#) (normally `mail`),
- [ldap_alias_addresses](#) (normally `mailAlternateAddress`),
- [ldap_equivalence_addresses](#) (normally `mailEquivalentAddress`),

- [ldap_personal_name](#),
- [ldap_capture](#),
- [ldap_recipientlimit](#),
- [ldap_recipientlimit](#),
- [ldap_sourceblocklimit](#),
- [ldap_preferred_language](#) (normally preferredLanguage),
- [ldap_source_conversion_tag](#) (as of JES MS 6.2),
- [ldap_blocklimit](#) (as of JES MS 6.3) (normally mailMsgMaxBlocks),
- [ldap_source_channel](#) (as of JES MS 6.3),
- [ldap_source_optinn](#) (as of JES MS 6.3),
- [ldap_preferred_country](#) (as of JES MS 6.3), and
- [ldap_spare_n](#) (as of JES MS 6.3-0.15).

The recommended setting for the [reverse_url](#) MTA option also uses the [\\$V substitution](#) for locating the domain in which the sending user address is located. Because of this implied lookup of the sending user's domain, the MTA's message processing can then also make use of per-sending-domain LDAP attributes including those named by the MTA options:

- [ldap_domain_attr_report_address](#) (normally mailDomainReportAddress),
- [ldap_domain_attr_blocklimit](#) (normally mailDomainMsgMaxBlocks),
- [ldap_domain_attr_recipientlimit](#),
- [ldap_domain_attr_recipientcutoff](#),
- [ldap_domain_attr_source_conversion_tag](#),
- [ldap_domain_attr_sourceblocklimit](#), and
- [ldap_domain_attr_source_channel](#).

Note: In actual operation, the MTA and domain map caching of domain lookup results means that the domain attributes are often available from a cache, without need for an additional actual LDAP query at this point. That is, while the [reverse_url](#) caused fetching of the sending user's personal LDAP attributes is relatively likely to involve a query all the way to the backend LDAP server, the "fetching" of the sending user domain LDAP attributes is often short-circuited, with the domain attributes cached due to a prior lookup.

So the list of potential side-effects resulting from address reversal, when it is properly configured to fetch these various per-sending-user and per-sending-domain LDAP attributes, is quite extensive, including effects on message size limits, message recipient limits, conversion tags, message capture, spam/virus filter processing opt-in, archiving opt-in, source channel "switching", and (if a notification message must be generated), notification language preference, non-return-of-content in notification messages, and per-domain postmaster address selection, *etc.*

New in the 8.0 release, bits of the [use_reverse_database](#) MTA option can be set to disable use of either the envelope From (MAIL FROM) address, or the authenticated sender address, for purposes of source-based message size or recipient limit settings, as well as capture actions.

35.12.2 Reverse database

During the [address reversal](#) stage of address processing (which note occurs after rewriting via the MTA's [rewrite rules](#)), first any [reverse_url](#) LDAP-based address reversal is performed, as discussed in [LDAP lookups for address reversal](#). After any such LDAP-based address reversal, then header From: addresses and other backwards-pointing addresses and forwards-pointing header addresses may receive yet another address reversal processing step which makes use of the address reversal database and [REVERSE mapping](#).⁹

The relevance of the address reversal database is quite limited nowadays, as nowadays the sorts of changes it used to be used to make are instead performed via [LDAP lookups for address reversal](#). However, the process of its use will be described below.

When use of the address reversal database has been configured, the MTA uses each address specification, with any routing address but less any personal name fields, as an index key to the special database called the *reverse database*.¹⁰

Note that the format of probes to the [reverse database](#) (and to the [REVERSE mapping table](#)) can be affected by the [use_reverse_database](#) MTA option.

If the address is found in the reverse database, the corresponding right hand side from the database is substituted for the address.

If the address is not found, then an attempt is made to locate a mapping table named [REVERSE](#). No substitution is made and rewriting terminates normally if the table does not exist or no entries from the table match. But if the address does match a [REVERSE mapping](#) entry, then the result of the mapping is tested. The resulting string will replace the address if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. If the mapping entry specifies `$D` in addition to `$Y`, the resulting string will be run through the reversal database once more, and if a match occurs the template from the database will replace the mapping result (and hence the address).

Note that you do not need to have an address reversal database in order to use a [REVERSE mapping](#). That is, you can use a [REVERSE mapping](#) without having an address reversal database. And, of course, the reverse is true: you do not need to have a [REVERSE mapping](#) to use an address reversal database. Prior to the implementation of [LDAP lookups for address reversal](#), this back-and-forth consultation of reverse database, [REVERSE mapping](#), optionally reverse database again, was intended to allow convenient use and combination of the strengths of each facility: the reverse database's ability to make changes exactly targeted to a single address, and the [REVERSE mapping](#)'s ability to make generic, pattern-based changes to addresses. Nowadays, typically changes targeted to a single address are made via the LDAP entry for the user with that address, superceding former uses of the address reversal database, and even the [REVERSE mapping table](#) tends to get used only for special circumstances.

Entries in the address reversal database consist of two e-mail addresses: the address to match against and the address with which to replace a match. The database is usually created by preparing a text file and processing it with the `imsimta crdb` utility.

For example, suppose a site wishes to replace all reverse pointing addresses of the form `user@domain.com` with an address of the form `first.last@domain.com` where `first.last` is formed from the first (given) and last (family) names of the owner of the account `user`. This will then cause the outside world to only see addresses of the form `first.last@domain.com` and never see internal addresses. A text file `reverse.txt` containing lines of the form

```
user1@domain.com  first1.last1@domain.com
user2@domain.com  first2.last2@domain.com
.                 .                 .                 .
```

could then be set up and converted to an address reversal database with the UNIX commands,

```
# imsimta crdb reverse.txt IMTA_ROOT:data/db/reversedb-tmp
```

```
# imsimta renamedb IMTA_ROOT:data/db/reversedb-tmp IMTA_ROOT:data/db/reversedb
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated.

As another example, suppose that the internal addresses at domain.com are actually of the form user@hostX.domain.com, but, fortunately, the username space is such that user@hosta.domain.com and user@hostb.domain.com specify the same person for all hosts at domain.com. Then, rather than have to enter all possible user and host combinations in the address reversal database, the following, very simple [REVERSE mapping](#) may be used in conjunction with the address reversal database:

```
REVERSE
```

```
*@*.domain.com          $0@domain.com$Y$D
```

This mapping maps addresses of the form user@host.domain.com to user@domain.com. The \$D flag causes the address reversal database to then be consulted. The address reversal database should contain entries of the form shown in the previous example.

Although there is no address reversal database or [REVERSE mapping table](#) by default, their use for address reversal is activated automatically once such an address reversal database (depending upon the [use_reverse_database](#) MTA option value) or [REVERSE mapping](#) exists.

⁹ Address reversal processing can be restricted to only backwards pointing addresses if the third bit, bit 2, in the MTA option [use_reverse_database](#) is cleared. Application of this processing to envelope From address can be disabled using the [reverse_envelope](#) MTA option. The [noreverse](#) channel option can disable address reversal from being performed during enqueues to particular destination channels. However, note that at typical Messaging Server sites such options should **not** be used -- address reversal should **not** be disabled -- as a wide range of functionality, including functionality that might not at first glance seem to be address reversal related, depends critically upon normal address reversal processing.

¹⁰ Depending upon the setting of the MTA option [use_text_databases](#), the reverse "database" is either stored and accessed as an on-disk database (the default), or as an in-memory structure constructed (during configuration compilation or MTA initialization) from an on-disk flat text file. Or new in MS 8.0, the reverse "database" can be stored in memcache; see the [reverse_database_url](#) MTA option. The on-disk database, if that is what is being used, used to be located via the (now deleted) [imta_reverse_database](#) MTA Tailor option; nowadays its location is simply `IMTA_ROOT:data/db/reversedb`. This database file is built with the `imsimta crdb` utility from some site-supplied source text file, and the database itself must be world-readable for proper operation. If an in-memory database structure is instead being used, then when the MTA configuration is compiled (or at MTA process initialization time, if a compiled configuration is not in use) the MTA reads the file `IMTA_ROOT:config/reverse.txt` (formerly relocatable via the [imta_reverse_data](#) MTA Tailor option) and compiles it into an in-memory structure. This file should be world-readable for proper operation. Use of an in-memory "database" is normally recommended (for reasons of performance and reliability); however, do note that use of this in-memory "database" does require recompiling and reloading the configuration to get changes to the "database" (changes to the source text file) incorporated into the active configuration.

35.12.3 REVERSE mapping table

During the [address reversal](#) stage of address processing (which note occurs after rewriting via the MTA's [rewrite rules](#)), first any [reverse_url](#) LDAP-based address reversal is performed, as discussed in [LDAP lookups for address reversal](#). After any such LDAP-based address reversal, then header From: addresses and other backwards-pointing addresses and forwards-pointing header addresses may receive yet another address reversal processing step which makes use of the [address reversal database](#) and `REVERSE` [mapping table](#).⁹

Nowadays, the [reverse database](#) is very little used, having mostly been superceded for general address reversal purposes by the use of [LDAP lookups for address reversal](#); the `REVERSE` mapping table is also seldom needed or used for general address reversal purposes nowadays, but does sometimes get used under special circumstances. Thus while in principle the reverse database and `REVERSE` mapping can apply in an alternating fashion -- see the discussion of the [reverse database](#) for details -- this discussion of the `REVERSE` mapping table will focus on the `REVERSE` mapping table alone or as an adjust to LDAP lookups for address reversal. (Note that you do not need to have an [address reversal database](#) in order to use a `REVERSE` mapping table. That is, you can use a `REVERSE` mapping without having an address reversal database. And, of course, the reverse is true: you do not need to have a `REVERSE` mapping to use an address reversal database.)

After the other address reversal mechanisms have applied ([LDAP lookups for address reversal](#) and the [reverse database](#)), the MTA checks for whether a `REVERSE` mapping table exists. If a `REVERSE` mapping table does exist, the MTA will probe the mapping table with, by default, simply the current (as already reversed by other mechanisms) address. Note that the exact format of probes to the `REVERSE` mapping table (and [reverse database](#)) can be affected by the [use_reverse_database](#) MTA option, which can cause inclusion of channel names in the probe and as of MS 8.0, even the header field from which the address was taken. And as of MS 7.0.5, probes to the `REVERSE` mapping table can also be affected by the [include_conversiontag](#) MTA option.

If the address probe matches a `REVERSE` mapping entry, the result of the mapping is tested. The resulting string will replace the address if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. (If the mapping entry specifies `$D` in addition to `$Y`, the resulting string will be run through the [reversal database](#) once more, and if a match occurs the template from the database will replace the mapping result, and hence the address.)

New in MS 7.0u1, the output (template) of the `REVERSE` mapping is interpreted as a series of addresses separated by commas. As always, the first address becomes the reversal result if the entry sets the `$Y` flag; but new in MS 7.0u1, if the `$H` flag is also set and the input to the `REVERSE` mapping was the MAIL FROM (envelope From) address, then the second address in the comma-separated list becomes the default postmaster address for this sender.

See [Table of REVERSE mapping table flags](#) for a description of additional flags available for the `REVERSE` mapping, and [Mapping template substitutions and metacharacters](#) for a list of general mapping table substitution sequences and metacharacters.

Table 35.3 `REVERSE` mapping table flags

Flags	Description
<code>\$Y</code>	Use output as new address
<code>\$N</code>	Address remains unchanged
<code>\$D</code>	Run output through the reversal database
<code>\$A</code>	Add pattern as reverse database entry

\$F	Add pattern as forward database entry
\$H	(New in MS 7.0u1) If the address input to the REVERSE mapping is the envelope From (MAIL FROM address for SMTP submissions) then use the second address (out of the list of comma-separated addresses in output) as the default postmaster address for this sender
\$I	(New in MS 7.0) Consider address to have matched for this reversing purpose
\$G	(New in MS 7.0) Consider address not to have matched for this reversing purpose
Flag comparisons	Description
\$:B	Match only header (body) addresses
\$/B	Match only if not a header (body) address
\$/C	(New in MS 7.0u1) Match only if this probe is attempting to produce a canonical address for MTA use in comparison operations
\$/E	Match only envelope addresses
\$/E	Match only if not an envelope address
\$/F	Match only forward pointing addresses
\$/F	Match only if not a forward pointing address
\$/M	(New in MS 7.0u1) Match only if this probe is attempting to produce a reversed MAIL FROM address in address validity checks when the canonical form has not been selected
\$/R	Match only backwards pointing addresses
\$/R	Match only if not a backward pointing address
\$/I	Match only message-ids; see Internal host names in Received: and Message-Id: header lines for an example
\$/I	Match only if not a message-id

Note that if you have a compiled configuration, then you must recompile and reload your configuration in order for changes to the REVERSE mapping table (or indeed changes to any mapping table) to take effect.

⁹ Address reversal processing can be restricted to only backwards pointing addresses if the third bit, bit 2, in the MTA option [use_reverse_database](#) is cleared. Application of this processing to envelope From address can be disabled using the [reverse_envelope](#) MTA option. The [noreverse](#) channel option can disable address reversal from being performed during enqueues to particular destination channels. However, note that at typical Messaging Server sites such options should **not** be used -- address reversal should **not** be disabled -- as a wide range of functionality, including functionality that might not at first glance seem to be address reversal related, depends critically upon normal address reversal processing.

35.12.4 Subaddresses and address reversal

New in 7.0.5, the MTA's [address reversal](#) logic has been extensively redesigned to improve the handling of subaddresses. Previously the presence of a subaddress would prevent address reversal from occurring. (This long-standing behavior was a remnant of the past when if a user was sophisticated enough to put on a subaddress, one could presume that the user was sophisticated enough to have already specified the exact address that they wanted to send

from---so altering such an address wouldn't be necessary and indeed would be dubious. However, nowadays many other behaviors and [side-effects](#) are triggered via address reversal so matching regardless of subaddress is typically desirable, and further the old assumption that reversal is no longer desired in such cases is no longer as likely.)

As of 7.0.5, the default behavior will be to attempt to match the address with or without the subaddress. If there's a match, then the subaddress will be transferred to any rewritten address. This behavior may be explicitly specified by setting the [subaddressrelaxed](#) channel option (the default) on the source channel. [subaddresswild](#), if set, will match against subaddresses but disables transfer of the subaddress to the rewritten address. Finally, [subaddressexact](#) disables special subaddress handling during the reversal process.

35.12.5 RFC 822 comment strings and personal name modification

While not strictly an issue of [address reversal](#), a related topic is that of modifying the [RFC 822](#) phrase (more commonly referred to as a "personal name") or [RFC 822](#) comment that may appear associated with an address in a header line. Phrases appear, possibly quoted depending on the contents, before a format address specification (where the address specification is enclosed in angle brackets); comments are text appearing within parentheses. For instance, in

```
"John Q. Doe" <John.Doe@acme.com> (V.P. of Widget Development)
```

the "John Q. Doe" is an [RFC 822](#) phrase, that is, personal name, and the (V.P. of Widget Development) is a comment.

The MTA only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process personal names (strings preceding angle-bracket-delimited addresses) and comment strings (strings enclosed in parentheses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

In direct LDAP mode, the LDAP attribute named by the [ldap_personal_name](#) MTA option if present, will be used as the personal name in an address. (Note that any eight bit characters in the value will be assumed to be UTF-8 and be encoded as such.) The MTA will quote, if appropriate, the personal name value obtained from LDAP, according to [RFC 822](#) rules for such quoting; implemented as of JES MS 6.2 for normal messages, or as of JES MS 6.2p6 when generating messages such as vacation messages.

There are a number of channel options controlling the MTA's optional removal or modification of personal names and comment strings. This section will talk in detail about the [personalmap](#), [sourcepersonalmap](#), [commentmap](#), and [sourcecommentmap](#) channel options used for triggering general, mapping table based modifications to such strings; see the [personal * channel options](#) and the [comment * channel options](#) for a complete list of additional options including channel options appropriate when you simply wish to strip off all such strings.

When the `personalmap` keyword is present on a destination channel, then the MTA will run any personal names appearing associated with addresses in addressing header lines (*e.g.*, To:, Cc:, *etc.*, sorts of header lines) and message id header lines through a `PERSONAL_NAMES`

mapping table, if such a table exists. This is performed after any [address reversal](#). If no such table exists, then `personalmap` is equivalent to `personalstrip`; see the `personalstrip` channel option. The `sourcepersonalmap` keyword acts analogously for header lines on incoming messages; that is, it applies to source channels. The probe to the `PERSONAL_NAMES` mapping table by default takes the form

```
personal-name | address
```

or if (new in JES MS 6.1) the MTA option `use_personal_names=1` is set, then the probe takes the form

```
source-channel | destination-channel | personal-name | address
```

If the probe matches a mapping entry, the result of the mapping is tested. The resulting string will replace the personal name if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. Note that any eight bit values in the result will be assumed to be in the UTF-8 charset, and encoded as such. As of JES MS 6.2p3, the MTA will quote the result of the mapping, if appropriate according to the personal name quoting rules specified in [RFC 822](#). See [Table of PERSONAL_NAMES mapping table flags](#) for a description of additional flags available for the `PERSONAL_NAMES` mapping, and [Mapping template substitutions and metacharacters](#) for a list of general mapping table substitution sequences and metacharacters.

Table 35.4 PERSONAL_NAMES mapping table flags

Flags	Description
<code>\$Y</code>	Use output as new personal name
<code>\$N</code>	Personal name remains unchanged
Flag comparisons	Description
<code>\$:F</code>	Match only forward pointing addresses
<code>\$;F</code>	Match only if not a forward pointing address
<code>\$:R</code>	Match only backwards pointing addresses
<code>\$;R</code>	Match only if not a backwards pointing address
<code>\$:I</code>	Match only message-ids
<code>\$;I</code>	Match only if not a message-id

An example of a `PERSONAL_NAMES` mapping table, used in conjunction with `personalmap` on an appropriate channel, to cause addition of the `cn` LDAP attribute's value as a personal name only when no personal name was already present on an address would be:

```
PERSONAL_NAMES
```

```
! If a personal name is already present, use it as-is
!
! %* | *      $N
!
! When no personal name is already present, look up the
! domain in the address and determine whether it is one
! of "ours":
!
```

```

|*.*   $CBDN|$0@$1|$}$1, _base_dn_{
!
! If the domain was found, we're now probing with
! BDN|address|base-DN-for-users
!
BDN|*|*   \
$C$]ldap:/// $1?cn?sub?(|(mail=$0)(mailEquivalentAddress=$0))[$Y$E

```

When the `commentmap` channel option is present on a destination channel, then the MTA will run any comment strings appearing associated with addresses in addressing header lines (e.g., To:, Cc:, etc., sorts of header lines) and message id header lines through a `COMMENT_STRINGS` mapping table, if such a table exists. This is performed after any [address reversal](#). If no such table exists, then `commentmap` is equivalent to `commentstrip`; see the `comment*` channel options. The `sourcecommentmap` channel option acts analogously for header lines on incoming messages; that is, it applies to source channels. The probe to the `COMMENT_STRINGS` mapping table by default takes the form

```
comment-string | address
```

or if (new in JES MS 6.1) the MTA option `use_comment_strings` is set, then the probe takes the form

```
source-channel | destination-channel | comment-string | address
```

If the probe matches a mapping entry, the result of the mapping is tested. The resulting string (which should include enclosing parentheses) will replace the comment string if the entry specifies a `$Y`; a `$N` will discard the result of the mapping. See [Table of COMMENT_STRINGS mapping flags](#) for a description of additional flags available for the `COMMENT_STRINGS` mapping, and [Mapping template substitutions and metacharacters](#) for a list of general mapping table substitution sequences and metacharacters.

Table 35.5 COMMENT_STRINGS mapping table flags

Flags	Description
<code>\$Y</code>	Use output as new comment string
<code>\$N</code>	Comment string remains unchanged
Flag comparisons	Description
<code>\$:F</code>	Match only forward pointing addresses
<code>\$:R</code>	Match only backwards pointing addresses
<code>\$:I</code>	Match only message-ids

When thinking about personal names and comment strings in address header lines, note that as of 7.5 the MTA supports private modifiers to the [Sieve "address" test](#), `:"display"` and `:"comment"`, to access the personal name and comment string, respectively.

35.13 Forwarding mail

The term *alias* often encompasses two separate types of functionality: address routing (which inherently relates specifically to envelope To addresses), and cosmetic changes to other instances of addresses (envelope From addresses, and header addresses). The cosmetic changes of [address reversal](#) do not apply per se to envelope To addresses. Rather, envelope

To addresses are continuously rewritten and modified as messages proceed through the mail system. The entire goal of mail 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 addition to the transformations of domain names available via rewrite rules and domain aliases, which are generally (though not necessarily) applied to all addresses in the domain, including instances of envelope To addresses, envelope To addresses may also be modified on a per-user basis via one or more mechanisms of mail forwarding.

The MTA provides several mechanisms for forwarding mail. The method appropriate to a task at hand depends upon the scope of the forwarding:

- *Forwarding mail for selected users.* To forward mail for selected users, it is best to use aliases. You may also use aliases to accept mail for a non-existent user and forward it on to one or more real users. See [Forwarding via user LDAP attributes](#), and [alias options](#).
- *Forwarding mail to a list of users.* Aliases are also used to create [mailing lists](#).
- *Forwarding mail for selected users in other than the local domain.* To forward mail for selected users in an arbitrary domain (a domain other than the local channel name), the best approach may depend on how the users are provisioned. For users provisioned via [alias options](#), use of a rewrite rule matching the domain to the local channel and alias lookups that include the domain name and that have a fall-through entry (see the [alias_domains](#) MTA option) may be appropriate. For users provisioned in LDAP, in the general case, modifying those users' LDAP entries to have [appropriate LDAP attributes for forwarding](#) is most explicit.
- *Pattern-matching the users for forwarding.* In the special case where a set of users whose mail is to be forwarded can be detected via simple string pattern matching, and where the forwarding to be performed requires only a simple string transformation, use of a domain catchall mapping on an LDAP-provisioned domain may be convenient; see the [ldap_domain_attr_catchall_mapping](#) MTA option.
- *Forwarding all mail for a given host to another host.* In this case there are several approaches. The most efficient method requires that you be able to blindly change `user@old-host` into `user@new-host` without any conflict in user names; *i.e.*, not have to worry that the username "user" on old-host conflicts with a different person on new-host who has the same username. When this is the case, simple MTA [rewrite rules](#) may be used. The less efficient, but just as effective, approaches involve using either a [FORWARD mapping table](#), [forward database](#), or [alias lookups](#). Or for domains provisioned in LDAP, in some cases use of domain-level LDAP attributes may be appropriate: see the [ldap_domain_attr_smarthost](#) MTA option (LDAP attribute `mailRoutingSmartHost`) and [ldap_domain_attr_routing_hosts](#) MTA option (LDAP attribute `mailRoutingHosts`).
- *Complicated rule-based forwarding.* For performing complicated, rule-based forwarding, use of a Sieve filter to perform [Sieve "redirect" actions](#) allows for great flexibility; such a Sieve filter may be configured at various levels, including [domain-level or user-level](#).

The MTA's [forward database](#) and/or [FORWARD mapping table](#), and domain catchall mapping tables (see the [ldap_domain_attr_catchall_mapping](#) MTA option) may be used for special sorts of forwarding purposes, such as pattern based forwarding, source-specific forwarding, or "autoregistration" of addresses. Note that the forward database and FORWARD mapping table, as well as domain catchall mappings, are intended for use primarily for these

special sorts of address forwarding; most sorts of address forwarding, however, are better performed using one of the MTA's other forwarding mechanisms.

35.13.1 Forwarding via user LDAP attributes

To forward the mail of a user provisioned in LDAP, the most straightforward approach is to set the value `forward` as a value of the user's `mailDeliveryOption` LDAP attribute, and then set one or more `mailForwardingAddress` LDAP attributes on the user entry with each having a value consisting of an address to which to forward the user's mail. (Note that the mentioned value "forward" and these mentioned LDAP attributes are configurable via MTA options: see the [delivery_options](#), [ldap_delivery_option](#), and [ldap_forwarding_address](#) MTA options, respectively.)

Note that forwarding a user's mail, and delivering the mail locally, are not mutually exclusive options: a user may have multiple values of `mailDeliveryOption`.

35.13.2 FORWARD mapping table

The `FORWARD` mapping table provides functionality of pattern-based forwarding (analogous to the way that the [REVERSE mapping table](#) provides pattern-based changes to non-routing addresses), and the `FORWARD` mapping table also provides a mechanism for source specific forwarding. If a `FORWARD` mapping table exists, it is applied to each envelope `To` address. The probe of the `FORWARD` mapping table by default consists simply of the current envelope `To` address:

```
envelope-to
```

But bits of the [use_forward_database](#) MTA option and [include_conversiontag](#) MTA option control inclusion of additional fields in the probe. Bit 4 (value 16) of the [use_forward_database](#) MTA option controls including the `source-channel` and `from-address` in the probe; bit 6 (value 64) of [use_forward_database](#) controls including the current `destination-channel` in the probe; new in JES MS 6.3, enabling bit 2 (value 4) of the [include_conversiontag](#) MTA option causes any current [conversion tags](#) on the message to be included in a comma-separated list clause in the probe. With all of the these additional fields enabled, the probe has the form

```
source-channel | from-address | destination-channel | tag-list | envelope-to
```

New in MS 8.0, the [include_mtpriority](#) MTA option and [include_spare2](#) MTA option control, respectively, the inclusion of `MT-PRIORITY` and expected message size, and LDAP "spare" attribute values, in the probe. Also new in MS 8.0 are new bits of the [use_forward_database](#) MTA option controlling inclusion of the initial form and intermediate form of the recipient address in the probe.

Note that when the `from-address` is included in the probe, then the MTA options [use_orig_return](#), and (new in MS 6.3) [use_canonical_return](#), and (new in MS 7.0) [use_auth_return](#), can be used to select which form of the envelope `From` address is included.

If the probe matches a `FORWARD` mapping table entry pattern, the result of the mapping is tested. The resulting string will replace the envelope `To` address if the entry template specifies a `$Y`; a `$N` will discard the result of the mapping. See [FORWARD mapping table flags](#) for a list of additional flags, and see [Mapping tables](#) for general background and syntax of mapping

tables. If no entries in the FORWARD mapping table match, or if no FORWARD mapping table exists, then the MTA's envelope To address processing proceeds to its next stage.

The FORWARD mapping, if present, is consulted before any [forward database](#) lookup. If a FORWARD mapping matches and has the flag \$G, then the result of the FORWARD mapping will be checked against the forward database, if forward database use has been enabled via the appropriate setting of [use_forward_database](#). (Note that if channel specific forward database 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 database.) If a matching FORWARD mapping entry specifies \$D, then the result of the FORWARD mapping (and optional forward database lookup) will be run through the MTA's 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).

Table 35.6 FORWARD mapping table flags

Flags	Description
\$Y	Use output as new address
\$N	Address remains unchanged
\$D	Run output through the rewriting process again
\$G	Run output through the forward database , if forward database use has been enabled
\$S	Set the "trust subaddress as folder" flag, as if a folder name had been specified in a Sieve "fileinto" action
\$/	(New in MS 7.0u2) In an entry with \$Y\$D also set, treat the forwarding result as a mailing list with the original envelope From as the reporting address. This is similar in functionality to the specification of a / as the value of an <code>mgrpErrorsTo</code> value or a / as the value of a [ENVELOPE_FROM] named parameter in the alias file or alias database.
\$H	Disable further forward database or FORWARD mapping lookups
\$I	Hold the message as a .HELD file
\$V	(New in MS 8.0) Do not set the internal "alias match" flag, (normally as of MS 7.0u2 set by a matching FORWARD entry, so that viaaliasrequired is satisfied); that is, \$V means that viaaliasrequired is <i>not</i> satisfied by this FORWARD match
Flag comparisons	Description
\$:E	(New in MS 6.3) Incoming connection used ESMTP/EHLO
\$/E	(New in MS 6.3) Incoming connection did not use ESMTP/EHLO
\$:L	(New in MS 6.3) Incoming connection used LMTP/LHLO
\$/L	(New in MS 6.3) Incoming connection did not use LMTP/LHLO
\$:F	(New in MS 6.3) NOTIFY=FAILURES active for this recipient
\$/F	(New in MS 6.3) NOTIFY=FAILURES not active for this recipient
\$:S	(New in MS 6.3) NOTIFY=SUCSESSES active for this recipient
\$/S	(New in MS 6.3) NOTIFY=SUCSESSES not active for this recipient
\$:D	(New in MS 6.3) NOTIFY=DELAYS active for this recipient

\$;D	(New in MS 6.3) NOTIFY=DELAYS not active for this recipient
\$:A	(New in MS 6.3) SASL (SMTP AUTH) used to authenticate connection
;\$A	(New in MS 6.3) SASL (SMTP AUTH) not used
\$:T	(New in 6.3) SSL/TLS used to secure connection
;\$T	(New in 6.3) SSL/TLS not used
\$:P	(New in MS 7.0) Match only if POP-before-SMTP was used
;\$P	(New in MS 7.0) Match only if POP-before-SMTP was not used

35.13.3 Forward database

For cases where address forwardings need to be autoregistered, or other cases of source specific forwarding, the MTA's forward database is available. Note that use of the forward database for simple forwarding of messages is generally not appropriate: if a database must be used, then the alias database is more efficient for straightforward forwarding; or if users are provisioned in LDAP, then forwarding can be handled via [normal LDAP attributes](#). So by default, the forward database is not used at all; its use must be explicitly enabled via the [use_forward_database](#) MTA option.

Forward database lookups are performed after address rewriting and after alias expansion is performed, and after any [FORWARD mapping](#) is checked. If a forward database lookup succeeds, the resulting substituted address is then run through the MTA's address rewriting process all over again.

The forward database is traditionally an MTA `crdb` database, created using the [imsimta crdb utility](#) from a source text file. However, nowadays the `crdb` step is typically omitted by using an MTA so-called "text database": with the relevant bit (bit 2/value 4) of the [use_text_databases](#) MTA option set, then the MTA will compile the source text file directly into its configuration. In either case, 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 database has been enabled by setting bit 3 of the [use_forward_database](#) MTA option, then the source text file format expected is:

```
source-channel | source-address | original-addresschanged-address
```

For instance, suppose a `tcp_snads` channel receives messages from a gateway to a SNADS system that provides (and can use) only shortform hostnames. Then an entry such as

```
tcp_snads | Bobby@BLUE | 12345678@MSMTA      user.with.a.long.name@else.where.com
```

would allow the Bobby@BLUE SNADS user to send to a SNADS address of 12345678@MSMTA when they wish to send to user.with.a.long.name@else.where.com.



Chapter 36 Mailing lists

36.1 Mailing list addresses	36-1
36.1.1 Mailing list multiple access control interpretation	36-2
36.1.2 Password-protected mailing lists	36-3
36.1.3 Moderated mailing lists	36-3
36.2 Mass mailings	36-8
36.2.1 Defining membership of large lists	36-9
36.2.2 Proper use of lists rather than groups	36-16
36.2.3 Restricting posting access to large lists	36-19
36.2.4 Performance submitting mass mail messages	36-21
36.3 Special address formats for list members	36-22

The MTA has flexible mailing list facilities, and facilities for performing automated processing of certain (typically mailing list related) messages.

In the MTA, mailing lists are implemented as a special form of MTA alias: controlling mailing list headers, access to post to mailing lists, setting up moderated mailing lists, *etc.*, are controlled via the alias that defines the mailing list. See [Aliases](#) for background general information on aliases and their various implementation mechanisms, including [Direct LDAP aliases](#), [Unified Configuration alias options](#), the [MTA alias file](#), and the (optional) MTA alias database. Details specifically on mailing list definitions, with their many options and variations, are described in [Mailing list addresses](#).

See [Mass mailings](#) for a discussion of topics of particular interest in the case of sending announcements or emergency communications to very large numbers of users.

36.1 Mailing list addresses

A mailing list address is a special address created through the [alias file](#) or alias database, (or in Unified Configuration created as a set of values in an [alias group](#)), or stored as an [alias in a group entry in LDAP](#) ---see the discussion of [aliases](#) for general background on aliases, the alias file, and alias database, and LDAP lookups.

Associated with each mailing list address in the alias file or alias database or specified via a Unified Configuration alias option is a list of member addresses: that list of member addresses may be specified in the form of a text file which contains one or more mail addresses, or as an LDAP URL that returns one or more mail addresses. For a mailing list address stored in LDAP, the LDAP mail group with that address will have additional LDAP attributes specifying member addresses in any of a number of ways: attributes whose values specify member addresses directly, attributes whose values specify other user or group entries in LDAP (which should themselves be decorated with actual email addresses to use as member addresses), or attributes whose value is a "dynamic" LDAP URL specifying list member addresses in the LDAP directory via filter criteria. Note that regardless of whether an LDAP query URL is specified for an alias in the alias file, alias database, or an alias option, or is specified as the value of a "membership" attribute on an LDAP group entry, the LDAP query URL may return multiple addresses either because the LDAP query matches multiple entries containing a desired attribute(s), or because the LDAP query matches a multivalued attribute of a single entry.

When a message is received by the MTA for a mailing list address, the message is then passed on to each address specified in the mailing list file or LDAP URL, or specified in "membership"

attributes of an LDAP mail group. Note that any such membership address may itself be an alias or another mailing list address.

For mailing lists defined via a group entry in LDAP, see the numerous LDAP attributes available for group entries as listed in [Table 7-4](#) with the MTA options that name them. And for defining the mailing list membership of mail group LDAP entries, see in particular the discussion of [Defining membership of large lists](#). For mailing lists defined in the alias file or alias database, see [Alias file mailing list aliases](#).

36.1.1 Mailing list multiple access control interpretation

Note that when specifying multiple sorts of posting access control parameters on a [mailing list address](#) (or other address), the effect is normally cumulative (a logical AND operation). For instance, specifying both [alias_cant_list](#) and [alias_auth_list](#) (in legacy configuration, [\[CANT_LIST\]](#) and [\[AUTH_LIST\]](#)) on a list normally means that only those addresses that are in the [alias_auth_list](#) and not in the [alias_cant_list](#) may post to the list. But the interpretation of combining access control settings may be altered via the [or_clauses](#) MTA option; that is, with [or_clauses=1](#), the interpretation defaults to a logical OR. The interpretation of combining individual access controls on mailing lists can also be controlled for individual access controls on individual lists by using the [alias_and](#) or [alias_or](#) alias options (in legacy configuration [\[AND\]](#) or [\[OR\]](#) [alias file named parameters](#)); the use of such an option causes subsequent access controls (up until another occurrence of [alias_and](#) or [alias_or](#)) to be interpreted as specified.

Note also that the [alias_auth_list](#), [alias_auth_mapping](#), [alias_cant_list](#), and [alias_cant_mapping](#) alias options (the [\[AUTH_LIST\]](#), [\[AUTH_MAPPING\]](#), [\[CANT_LIST\]](#), and [\[CANT_MAPPING\]](#) [parameters](#) in legacy configuration) provide a separate sort of control from [alias_moderator_list](#) and [alias_moderator_mapping](#) alias options (the [\[MODERATOR_LIST\]](#) and [\[MODERATOR_MAPPING\]](#) [parameters](#) in legacy configuration); they do different things and may be used effectively in conjunction. The [alias_auth_*](#) and [alias_cant_*](#) options control who can post at all; only addresses that make it through those access filters then get checked for the next question, namely the [alias_moderator_*](#) access filter, which controls whether the sender can post directly *vs.* whether their attempted posting is referred to [alias_moderator_address](#) ([\[MODERATOR_ADDRESS\]](#) in legacy configuration).

In the direct LDAP environment, multiple list access controls are again normally essentially cumulative (a short-circuited logical AND operation) between different types of controls, although multiple values for a single type of allow control are ORed. (That is, multiple values of `mgrpAllowedBroadcaster` are effectively ORed with each other. And similarly, multiple values of `mgrpAllowedDomain` are effectively ORed with each other.) Specifically, the value(s) of the attribute named by the [ldap_cant_url](#) MTA option (by default, `mgrpDisallowedBroadcaster`) is/are checked first, and if that check passes (the attempting poster does not match any `mgrpDisallowedBroadcaster` value) then next the value of the attribute named by the [ldap_auth_url](#) MTA option (by default, `mgrpAllowedBroadcaster`) is checked (ORing the possibilities if multiple `mgrpAllowedBroadcaster` values are specified; that is, an attempted poster who matches any of the `mgrpAllowedBroadcaster` values will be allowed), and if that check passes then next the value of the attribute named by the [ldap_cant_domain](#) MTA option (by default, `mgrpDisallowedDomain`) is checked, and if that check passes (the attempting poster does not match any `mgrpDisallowedDomain` value) then next the value of the attribute named by the [ldap_auth_domain](#) MTA option (by default, `mgrpAllowedDomain`) is checked (ORing the possibilities if multiple `mgrpAllowedDomain` values are specified; that is, an attempted poster who matches any of the `mgrpAllowedDomain` values will be allowed). So

this is essentially a "short-circuited" logical AND of the posting access restriction conditions.

But the interpretation of combining different types of access control settings may be altered via the [or_clauses MTA option](#); that is, with `or_clauses=1`, the interpretation defaults to a logical OR. Individual mailing lists or groups can override the general setting of the MTA option `or_clauses` via a value of "OR" or "AND" as one of the values of the attribute named by the [ldap_auth_policy MTA option](#) (by default, `mgrpBroadcasterPolicy`). Note that the configuration choice of combining different *types* of controls with OR *vs.* AND does *not* affect the interpretation of multiple values of a single type of control: for instance, multiple values of `mgrpDisallowedBroadcaster` are always ANDed together (a poster must pass all the conditions) while multiple values of `mgrpAllowedBroadcaster` are always ORed together (a poster may post if their address matches any of the conditions).

36.1.2 Password-protected mailing lists

Mailing lists may be set up to require a password in order to post. For mailing lists defined via the aliases file or alias database, the password value (or list of values) is specified via the [PASSWORD] mailing list named parameter; see [Alias file named parameters](#). In Unified Configuration, the analogous setting is the [alias_password](#) alias option. For mailing lists defined in LDAP, the password value (or list of values) is specified via the `mgrpAuthPassword` attribute (or more precisely, whatever attribute is specified by the [ldap_auth_password MTA option](#)).

In order for a message to be posted to the list, the MTA will then require that the message include, on an Approved: header line, one of the authorizing passwords. During the mailing list expansion process, the MTA will remove that password value from the Approved: header line, and indeed remove the entire Approved: header line if that was the only value present. (In the case of multiple values, all passwords for a list will be removed. So in the case of nested lists, make sure that the list password sets for the different lists are disjoint.) So note that the Approved: header line value that allowed the message to be posted will *not* be exposed in the actual posted message. Members of the list will not see the password. The group of users who can post (who know and use the password) can be a subset, or even a completely separate group of users.

Note that since a single mailing list can have multiple allowed passwords, it is possible to assign a separate password to each allowed poster.

Since mailing lists may be nested (one mailing list may be subscribed as a member of another mailing list), and since mailing list may have its own password(s), an Approved: header line may contain multiple, comma-separated values.

Password-protected mailing lists automatically get [deferred expansion of membership](#).

36.1.3 Moderated mailing lists

A *moderated mailing list*---a list where certain moderators are allowed to post messages, but attempted postings by others are routed to a moderator or moderators instead of being posted directly---requires three basic settings for moderation (in addition to the [mgrpErrorsTo](#) setting that makes it a true list, rather than a group). These are, for a list defined in LDAP:

1. The moderator(s) allowed to post directly must be among the [mgrpAllowedBroadcaster](#) values.
2. The [mgrpMsgRejectAction](#) must be set to `toModerator`, to cause messages that initially fail an access check to get routed to the moderator(s) (who may or may not resend the message to the list, at their pleasure) rather than being outright rejected.

3. The address(es) to which to re-route not-yet-approved direct posting attempts must be specified, by setting `mgrpModerator`.

(For concreteness, the above description refers to the LDAP attribute names that the MTA consults by default; but more precisely, the LDAP attributes that must be used are whatever attributes are named by the `ldap_errors_to`, `ldap_auth_url`, `ldap_reject_action`, and `ldap_moderator_url` MTA options.)

For a list defined via [alias options](#) instead of in LDAP, the required basic moderation settings, beyond the basic list setting of `alias_envelope_from`, are:

1. A moderator address, (which may be an alias/group/list address itself), must be set via the `alias_moderator_address` alias option; this is the address to which will be sent all attempted postings *not* coming from a moderator address. This is analogous to the `mgrpModerator` LDAP setting.
2. Optionally, additional addresses may be established as allowed direct posters (additional originator addresses to be handled as if they were moderators, for purposes of allowing postings) using the `alias_moderator_list` alias option (where in this specific case the use is rather similar to that of the `ldap_auth_url` LDAP setting), or any of the `alias_moderator_mapping`, `alias_username_moderator_list`, `alias_sasl_moderator_list`, or `alias_sasl_moderator_mapping` alias options. But, unlike the LDAP case, in the absence of any settings specifying additional accepted moderator From: addresses, the basic `alias_moderator_address` address will be used not only as the moderator address to which to *send* unmoderated postings, but also be recognized as a valid moderator From: address from which to *accept* direct postings.

Once these settings are made, a list can be considered moderated. Note that while a common case is that there is only *one* moderator and *only* that one moderator can post, other cases may be just as useful. In particular, allowing specially known and privileged classes of non-moderators as additional `mgrpAllowedBroadcaster` posters can be useful; or combining (ORing) various posting access conditions via `mgrpBroadcasterPolicy` so that moderation applies only to attempted postings that fail other conditions (such as use of SMTP AUTH to authenticate, inclusion of a list password, *etc.*) can allow for moderation of messages meriting higher scrutiny, while permitting more trusted senders to post directly.

For instance, it can be useful to set up a mailing list where postings from members of the list are not moderated, but attempted postings by non-members of the list go to the moderator for human review. To set up a mailing list where members are allowed to post directly, but where attempted postings by non-members must be approved by the list moderator, include in the list entry attributes such as:

```
mgrpErrorsTo: moderator-user@domain.com
mgrpAllowedBroadcaster: mailto:list-members@domain.com
mgrpAllowedBroadcaster: mailto:moderator-user@domain.com
mgrpMsgRejectAction: TOMODERATOR
mgrpModerator: mailto:moderator-user+listname@domain.com
mailDeferProcessing: AFTER_AUTH
```

Note that in the above example, the new in JES MS 6.3p1 `AFTER_AUTH` value is used for `mailDeferProcessing`. In earlier versions, the less desirable `no` value would need to be used instead.

A similar example would be a mailing list where messages containing the list password are posted directly, while message lacking that password are directed to the moderator for

inspection. By setting the `mgrpAllowedBroadcaster` to be the moderator, and setting `mgrpBroadcasterPolicy` to `PASSWORD_REQUIRED, OR`, the effect is that postings from the moderator will be allowed even without the normal password, while attempted postings from any other senders will only be directly posted if they contain the password.

```
mgrpErrorsTo: moderator-user@domain.com
mgrpBroadcasterPolicy: PASSWORD_REQUIRED, OR
mgrpAllowedBroadcaster: mailto:moderator-user@domain.com
mgrpAuthPassword: secret-password
mgrpModerator: mailto:moderator-user+listname@domain.com
mgrpMsgRejectAction: toModerator
```

When the moderator receives a message addressed to `moderator-user+listname@domain.com`, the moderator would either reject messages he/she does not approve, or resend the message to the list. In this example, due to the `mgrpBroadcasterPolicy` setting including `OR`, the moderator is not required to add the header line that other senders would need to include:

```
Approved: secret-password
```

That is, by setting the `mgrpAllowedBroadcaster` to be the moderator, and setting `mgrpBroadcasterPolicy` to `PASSWORD_REQUIRED, OR`, the effect is that postings from the moderator will be allowed even without the normal password, while attempted postings from any other senders will only be directly posted if they contain the password.

Many additional mailing list behaviors are best achieved by setting up one form or another of moderated list. This is true even for certain behaviors that may not immediately seem like cases of a moderated mailing list. An example: setting up a mailing list where postings directly to the list are permitted without moderation, but where replies to prior list postings -- attempts to continue a posting thread -- are disallowed can be achieved via a moderated list setup. Indeed in general, cases where it is desired to perform more elaborate checks or processing on messages before they are allowed to be posted to the list are often cases where appropriate moderation of the list may be helpful.

For instance, consider the case of a mailing list that disallows replies to prior list postings. Replies to prior list postings will be detected via a Sieve filter that looks for common "reply" prefix strings on the Subject: header line. The mailing list can be set up to include attributes:

```
mgrpErrorsTo: <hogwarts-moderator-user>@domain.com
mgrpModerator: mailto:<hogwarts-moderator-user>+hogwarts@domain.com
mailDeferProcessing: AFTER_AUTH
```

Then for the `<hogwarts-moderator-user>@domain.com` user, set up a Sieve filter that looks for messages addressed to the user with the subaddress `hogwarts` (use the [Sieve subaddress extension](#)) and then either does a 'reject' or a 'redirect :resetmailfrom "hogwarts@domain.com" ;', as appropriate. *E.g.*:

```
require ["reject", "envelope", "subaddress"];
if envelope :detail "to" "hogwarts" {
    if allof ( header :contains ["To", "Cc", "Bcc"] "hogwarts@domain.com",
              header :matches "Subject" ["Re:*:*" , "RE:*:*"]) {
```

```
    reject "Replies not allowed!"; }
  else { redirect :resetmailfrom "hogwarts@domain.com"; }
}
...whatever other filter stuff this user wants...
```

Another example where a technically moderated list, without involving actual human moderation, may be of interest would be the case of rejected attempted postings that include non-text parts. With a mailing list definition that includes:

```
mail: listname@domain.com
mgrpErrorsTo: listname-owner@domain.com
mgrpModerator: mailto:list-moderator+listname@domain.com
mailDeferProcessing: AFTER_AUTH
```

then the moderator user or pseudo-user (since possibly no human being corresponds to this entry, which might exist for the sole purpose of having this Sieve script), *list-moderator@domain.com*, might use a Sieve filter along the lines of:

```
require ["mime","reject","subaddress","envelope"];
if envelope :detail "to" "listname" {
  if header :mime :anychild :type "Content-type"
    ["application","audio", "image","video"]
    reject "Non-text may not be posted to this list";
  else { redirect :resetmailfrom "listname@domain.com"; }
}
...whatever other filtering this user wants...
```

An example that does involve an actual human moderator would be for a list that does not impose controls on the identity of posters, but where the attempted posts are scanned for suspicion of spam content, and then are potentially either rejected or human moderated at different levels of spam suspicion, while scanned non-spam messages get automatically posted.

```
mail: listname@domain.com
mgrpErrorsTo: listname-owner@domain.com
mgrpModerator: mailto:listname-owner+listname@domain.com
mgrpAllowedBroadcaster: mailto:listname-owner@domain.com
mgrpMsgRejectAction: toModerator
```

Then the *listname-owner@domain.com* "user":

```
mail: username@domain.com
mailEquivalentAddress: listname-owner@domain.com
```

should have a Sieve filter including:

```
require ["fileinto","reject","subaddress","spamtest","relational"];
if envelope :detail "to" "listname" {
  if spamtest :value "ge" :comparator "i;ascii-numeric" "10" {
    if spamtest :value "ge" :comparator "i;ascii-numeric" "20"
```

```

        { reject "Content appears to be spam"; }
    else
        { fileinto "listname-spam"; }
    } else { redirect "listname@domain.com"; }
}

```

Or another example of a list involving human moderator of *some* attempted postings would be where it is desired that rather than a list having an absolute limit on the size of postings (where note an absolute limit could be achieved simply by setting the desired `mgrpMsgMaxSize` on the list), that a moderator be able to selectively approve the posting of messages that would otherwise be considered "too large". Note that a size limit such as `mgrpMsgMaxSize` can not be simply ORed with access checks such as `mgrpAllowedBroadcaster`, so this goal will require a special moderation setup.

```

mail: listname@domain.com
mgrpErrorsTo: listname-owner@domain.com
mgrpModerator: mailto:listname-owner+listname@domain.com
mgrpAllowedBroadcaster: mailto:listname-owner@domain.com
mgrpMsgRejectAction: toModerator

```

Then the `listname-owner@domain.com` "user":

```

mail: username@domain.com
mailEquivalentAddress: listname-owner@domain.com

```

could have a Sieve filter including:

```

require ["reject", "subaddress", "fileinto", "notify"];
if envelope :detail "to" "listname" {
    if size :under 10K {
        redirect :resetmailfrom "listname@domain.com"; }
    elsif size :under 50K {
        fileinto "listname-large";
        notify :method "email" :options
            "listname-owner@domain.com"
            "Message filed to listname-large needs to be checked"
            "";
    }
    else { reject "Message too large to post"; }
}

```

which would automatically post to the list any small messages, automatically reject any extremely large messages, but notify the moderator of any new medium size message posting attempt and file that message into a `listname-large` folder for the moderator to inspect and approve or bounce, at their leisure.

Returning to yet another case of "automated moderator" processing, consider an example where it is desired to respond with an automatic message to list postings, using the Sieve "vacation" action. With a list entry along the lines of:

```

mail: listname@domain.com

```

```
mgrpErrorsTo: listname-owner@domain.com
mgrpModerator: mailto:moderator-pseudo-user+listname@domain.com
mgrpAllowedBroadcaster: mailto:moderator-pseudo-user@domain.com
mgrpMsgRejectAction: toModerator
mailDeferProcessing: AFTER_AUTH
```

Then the LDAP entry for *moderator-pseudo-user@domain.com*:

```
mail: moderator-pseudo-user@domain.com
```

can have a Sieve script attached along the lines of:

```
require ["subaddress","vacation"];
if envelope :detail "to" "listname" {
    vacation :addresses "listname@domain.com"
        :from "listname@domain.com"
        :subject "Welcome to listname!"
        "Thank you for your interest in listname.
You will receive a personal response soon.";
    redirect :resetmailfrom "listname@domain.com";
}
```

36.2 Mass mailings

The term *mass mailing* may be used to refer to cases of sending a certain message to all users hosted at a site, or to all users in some domain, or to all users in some organization unit, or to all members (including "external" members) of some "large" mailing list, *etc.*---any case where the number of recipients is relatively "large". The purpose of the message might be one of great urgency (such as an emergency communication), or it might be of general interest but low urgency (such as routine announcements).

Since the MTA supports LDAP filter based (so-called "dynamic") group and list definitions, it is straightforward to define a list or group to consist of all users meeting some criteria (any criteria that may be specified in an LDAP URL filter). See in particular the MTA's support of `memberURL` and `mgrpDeliverTo` attributes (or more precisely, those attributes named by the `ldap_group_url1` and `ldap_group_url2` MTA options) in LDAP group entries. Or for MTA alias file defined groups and lists, see the LDAP URL membership syntax discussed in [Alias file LDAP URL alias values](#). And more complex lists can be constructed, including criteria-based sets of locally-hosted members along with external members listed by address, or lists with "nested" definitions of sub-lists, or "overlapping" definitions/membership.

Groups and mailing lists are most often defined to make use of actual e-mail addresses: either directly as a list of e-mail addresses, or by defining the membership to be users who each have a canonical e-mail address. However, the MTA, via its new in JES MS 6.3 `ldap_url_result_mapping` MTA option (and whatever LDAP attribute `ldap_url_result_mapping` names), also supports defining groups and mailing lists for which e-mail addresses are constructed from other LDAP attributes that do not themselves contain proper e-mail addresses.

Furthermore, as of JES MS 6.3 and its new `process_substitutions` MTA option, it is possible to define "meta-groups" and "meta-lists": where a single meta-list definition provides what amounts to an entire collection of definitions of different lists.

As of 7.0.5 and its new [GROUP_AUTH mapping table](#), it is possible to use alternate LDAP attributes and values for group/list authorization checks. In particular, this can be useful when dealing with group/list information stored in an LDAP directory using a non-Oracle schema.

Now any time that a group or list with "large" membership of e-mail recipients is defined, and any time that a message is to be sent to an especially "large" number of recipients, there are some issues worth considering; these issues will each be discussed in greater detail in sections below.

- Defining the actual list membership; see [Defining_membership_of_large_lists](#).
- Sensible error handling; see [Proper use of lists rather than groups](#).
- Restrictions on senders; see [Restricting posting access to large lists](#).
- Performance in submitting the message; see [Performance submitting mass mail messages](#).
- Impact of this message (and possibly its multiple copies requiring processing) on other message processing; see [Addresses per message copy](#).
- The appropriate choice of processing priority for the message. This may vary from "urgent" for messages that are time-critical, to "non-urgent" for messages that while of general *interest* are not time-critical and might be more efficiently processed during "off hours". (Note that "importance" is a separate measurement than "processing priority": messages can be time-critical but not very important, as for instance a message that a party is about to start in the coffee room, or important without being time-critical, as for instance a message that system down-time is scheduled for two weeks away.)
- The timing of attempting delivery of the message: for some messages, it may be desirable to delay even *attempting* to deliver the message until some pre-determined time. See [SMTP SUBMIT FUTURERELEASE support](#).
- Efficient storage of the messages; see [Addresses per message copy](#).

36.2.1 Defining membership of large lists

Note: This discussion will focus on lists primarily defined in LDAP, as that is the typical mode of defining lists nowadays. However, the MTA does also support lists defined instead in an [aliases file](#) entry, or in Unified Configuration via [alias options](#), rather than in an LDAP entry; see those topics for details.

The membership of a list *may* be defined by simply listing the e-mail addresses of the members, or (for lists defined in LDAP) by listing the DNs (location in the LDAP directory tree) of the members.

For instance, for a list defined in LDAP, the `memberURL` attribute or `mgrpDeliverTo` attribute (more precisely, the attributes named by the `ldap_group_url2` and `ldap_group_url1` MTA options, respectively) are available, *e.g.*:

```
memberURL: ldap:///o=usergroup??sub?(uid=abc123)
memberURL: ldap:///o=usergroup??sub?(cn=Adam Brown)
memberURL: mailto:Betty.Charles@domain.com
memberURL: mailto:John.Doe@external-domain.com
```

Note that an LDAP based list definition can obtain a list of actual addresses from a separate, on-disk file (accessible from the MTA performing the list expansion) by using a [file: URL](#); *e.g.*:

```
memberURL: file:///IMTA_TABLE:list-members.txt
```

where the `IMTA_TABLE:list-members.txt` file itself consists of a list of e-mail addresses, one address per line, *e.g.*,

```
Betty.Charles@domain.com  
John.Doe@external-domain.com
```

Or, when it is more convenient to describe a list in LDAP via the locations of the member entries rather than via their e-mail addresses, the members may be described via the `uniqueMember` attribute (more precisely, the attribute named by the `ldap_group_dn` MTA option). For each member defined via `uniqueMember`, the MTA will use the value of that member's `mail` attribute (more precisely, the value of the attribute named by the `ldap_primary_address` MTA option) as the e-mail address for that member of the list. (The `group_dn_template` MTA option controls which attributes for the user the MTA fetches, and hence which/whether additional addresses for the user can match for purposes such as comparisons with posting restrictions. See [Indirect or alternate criteria for list membership](#) for a discussion of more complex membership definitions using variants of DN semantics.)

But for "large" lists especially, it is often more convenient to instead define the list membership in terms of some criteria by which to locate the relevant users in LDAP; for instance, a list of all users in LDAP might have membership defined as:

```
memberURL: ldap:///o=usergroup??sub?(&(objectClass=inetMailUser)  
                                         (objectClass=inetOrgPerson))
```

Or a list of all the users in the domain `domain.com` might have membership defined as:

```
memberURL: ldap:///dc=domain,dc=com,o=internet??sub?  
            (&(objectClass=inetMailUser)(objectClass=inetOrgPerson))
```

When defining list membership via a `memberURL` or `mgrpDeliverTo` attribute (or more precisely, whatever attributes to which the MTA options `ldap_group_url1` and `ldap_group_url2` are set), unless use of some other attribute is explicitly selected, the MTA assumes that the value of the `mail` attribute (`ldap_primary_address` MTA option) should be used as the e-mail address for the list member; note how in the above example no attribute is explicitly requested (and therefore the default use of the `mail` value is assumed). If it is desired to use some other attribute that does itself contain an e-mail address, that may be selected by explicitly selecting that attribute. For instance, suppose that a no longer canonical, acquired domain name is still in use in certain users' `mailEquivalentAddress` values; a list intended for sending to those users, at their "old" addresses, could have membership defined as:

```
memberURL: ldap:///o=usergroup?mailEquivalentAddress?sub?
  (&(objectClass=inetMailUser)
   (objectClass=inetOrgPerson)
   (mailEquivalentAddress=*@acquired-domain))
```

There is one additional factor to consider when defining list membership. Definitions that reference member e-mail addresses, *e.g.*, `mgrpRFC822MailMember` (see the [ldap_group_rfc822](#) MTA option), allow better error reporting in cases of definition errors (such as syntax errors), than definitions that reference LDAP DNs or LDAP URLs. Syntax errors in e-mail address members will be reported to the list owner. However, syntax errors in an LDAP DN or LDAP URL used to define members, *e.g.*, syntax errors in a `uniqueMembers` or `memberURL` attribute, will cause the list membership expansion to abort at that point. So it is especially important to check list definition, *e.g.*, via `imsimta test -rewrite -check_expansions`, when defining lists that make use of LDAP DN or LDAP URL criteria for membership.

36.2.1.1 Indirect or alternate criteria for list membership

As discussed in [Defining membership of large lists](#), the MTA's normal interpretation of the `uniqueMember` LDAP attribute (more precisely, the attribute named by the `ldap_group_dn` MTA option) involves expanding the value of the `uniqueMember` attribute via the URL template set by the `group_dn_template` MTA option, which by default is

```
ldap:/// $A??sub?(mail=*)
```

(meaning that the `$A substitution` inserts the `uniqueMember` value), so that by default, `uniqueMember` values are interpreted as specifying a DN location in the DIT: all e-mail addresses under that location are considered to have been specified (be members).

This sort of indirect, additional-step, expansion of an LDAP attribute value turns out to be potentially useful for alternate approaches for membership definition. In order not to interfere with the "normal" handling of `uniqueMember` DN values for list membership, in Messaging Server 7.0.5 the MTA option `ldap_group_dn2` and the mapping table `GROUP_TEMPLATES` were introduced. `ldap_group_dn2` can be used to specify the name of an LDAP attribute which will then be processed similarly to the `ldap_group_dn` (`uniqueMember`) attribute---in particular, by default values of the LDAP attribute named by `ldap_group_dn2` are expanded via the `group_dn_template` URL template just like `ldap_group_dn` values. But the real usefulness of `ldap_group_dn2` tends to be when its use is combined with use of the `GROUP_TEMPLATES` mapping table.

The `GROUP_TEMPLATES mapping table` provides a way to specify different URL expansion templates for differently named LDAP attributes (such as different templates for the attribute named by `ldap_group_dn` *vs.* the attributes named by `ldap_group_dn2`), or even for different values of such LDAP attributes. When a `GROUP_TEMPLATES` mapping table exists, it will be probed each time a group has an LDAP attribute named by either of the `ldap_group_dn` or `ldap_group_dn2` MTA options. The probe form is:

```
object-classes | attribute-name | attribute-value
```

where `object-classes` is a plus-separated list of the object classes associated with the current LDAP entry, (see the `ldap_group_object_classes` MTA option), `attribute-name` is the name of the group "DN" attribute being expanded (*i.e.*, the LDAP attribute name

specified for either `ldap_group_dn` or `ldap_group_dn2`), and `attribute-value` is that attribute's current value.

If the mapping sets the `$Y` output flag, then the mapping output string will be used as the template for this attribute's expansion in place of using the value of `group_dn_template` as the template. If the mapping sets the `$N` output flag, then the attribute will be silently ignored.

So now that the facilities have been explained, how could they actually be used? Well, one sort of usage might be where groups/lists are defined not so much by the group/list entry pointing to (that is, listing) the members, but rather where each user entry specifies the groups/lists of which the user is a member, referring to some group/list ID. For instance, suppose group/list entries have an LDAP attribute `listID` whose value is some string unique to that group/list. Then suppose further that user entries mark which groups/lists the user belongs to by having a `memberOf` attribute that contains a valid `listID` value. Defining group/list membership in this new way, while still allowing "traditional" `uniqueMember` membership definitions, can be achieved by configuring the MTA with an option:

```
msconfig> set mta.ldap_groupdn listID
```

and mapping table:

GROUP_TEMPLATES

```
! Normal use of ldap_group_dn attribute uniqueMember
*|uniqueMember|* $Yldap:/// $A?mail?sub?(mail=*)
! Find users who have a memberOf attribute set to the value of the group's
! listID attribute
*|listID|* $Yldap:/// $B??sub?(memberOf=$A)
```

where here note `$B` is the [substitution sequence](#) meaning to substitute in the base of the user/group portion of the DIT (the `ldap_user_root` MTA option's value), and where the `$A` "Address" substitution means, in this context, the value of the currently used LDAP attribute (so the value of, respectively, the `uniqueMember` or `listID` attribute in the respective mapping table entries matching those attribute names). Then to make use of this type of group/list definition, provision groups and users in the directory along the lines of:

```
group1-entry
listID: group123
...
group2-entry
listID: groupXYZ
...
user1-entry
memberOf: group123
memberOf: groupXYZ
...
```

36.2.1.1.1 Constructing list member addresses

Discussions of [Defining membership of large lists](#) and [Indirect or alternate criteria for list membership](#) have touched on potentially complex list membership criteria, as dynamic definitions may make use of complex LDAP search filters. However, in those discussions, the list addresses and member addresses themselves were not especially complex; the topic of complexity in address forms is the topic of this discussion. Via the MTA's support for manipulating the results of LDAP URL lookups, it is possible to define list membership in ways that "construct" e-mail addresses from non e-mail address attribute values, or that obtain e-mail addresses from "external" LDAP sources. And it is also possible to [define "meta-lists" where the list "address" itself is "dynamic"](#).

A list of SMS users, adding a domain name to an SMS ID value

For instance, let us assume a setting of

```
msconfig> set mta.ldap_url_result_mapping mgrpURLResultMapping
```

or in legacy configuration, in the MTA option file:

```
LDAP_URL_RESULT_MAPPING=mgrpURLResultMapping
```

Also assume that the MTA has an SMS channel configured with the domain sms.domain.com associated with it for some SMSC (that is, SMS service provider), and that each user who has an account with that SMS service provider has an smsID attribute containing their SMS ID, and that an MTA mapping table has been configured:

```
X-SMSID-TO-ADDRESS
* | *          "$1"@sms.domain.com$Y
```

Then a group sms-all@domain.com for the purpose of directing an SMS message to each such user could be defined via a group entry in LDAP that includes attributes:

```
mail: sms-all@domain.com
mgrpDeliverTo: ldap:///o=usergroup?smsID?sub
mgrpUrlResultMapping: X-SMSID-TO-ADDRESS
```

Adding different domain names to SMS IDs

At a site where users have various different SMS service providers, suppose with the domain for each service provider being stored in a SMSdomain attribute, then separate sub-lists for the different SMS service providers could be set up, with then a combined list for all the SMS users. Assuming also the following mapping table:

```
X-SMSID-DOMAIN-TO-ADDRESS
*(SMSdomain=*) | *          "$2"@$1$Y
```

as well as for each SMS service provider a group definition such as:

```
mail: sms1@domain.com
memberURL: ldap:///o=usergroup?smsID?sub?(SMSdomain=sms1-domain)
mgrpUrlResultMapping: X-SMSID-DOMAIN-TO-ADDRESS
```

Then a list of all SMS users could be defined via:

```
mail: sms-all@domain.com
mgrpErrorsTo: sms-errors@domain.com
memberURL: mailto:sms1@domain.com
memberURL: mailto:sms2@domain.com
...
```

36.2.1.1.1.1 Meta-group list definitions

In addition to manipulating LDAP attribute values to construct list member addresses as discussed in [Constructing list member addresses](#), the MTA also supports manipulating address forms to effectively create dynamic list names, via so called *meta-lists*.

A meta-list for per-department SMS users

As of JES MS 6.3 and its new [process_substitutions](#) MTA option, it is possible to define "meta-groups" and "meta-lists": to, for instance, pre-define via a single meta-list definition what amounts to an entire collection of different lists. For instance, with [process_substitutions=4](#) and [ldap_url_result_mapping=MgrpURLResultMapping](#) set,

```
msconfig> set mta.ldap_url_result_mapping mgrpURLResultMapping
msconfig# set mta.process_substitutions 4
```

then defining a list with attributes including

```
mail: sms@domain.com
mgrpErrorsTo: sms-errors@domain.com
memberURL: ldap:///o=usergroup?smsID?sub?(department=$S)
mgrpUrlResultMapping: X-SMSID-TO-ADDRESS
```

would make it possible to send an SMS message to every member of a given department who has an SMS account by sending to an address of the form `sms+department-value@domain.com`.

Members stored in an external LDAP directory

Another potential use of [ldap_url_result_mapping](#) to "construct" e-mail addresses would be where the member addresses are stored in an external LDAP directory, rather than in the usual user/group LDAP directory, with membership being defined in the external LDAP directory in an indirect fashion, via an `isMember` attribute. For instance, if the external directory stores the list definition along the lines of:

```
listName: extlist
```

```
listID: 1234abcd
listMemberRoot: dn-of-root-of-members
```

and then stores members of the list along the lines of:

```
mail: external-user1@domain.com
isMember: 1234abcd
```

then in the regular user/group LDAP directory define the list as:

```
mail: extlist@domain.com
mgrpErrorsTo: extlist-owner@domain.com
memberURL: extldap:///listname-search-base?listID?sub?(listName=extlist)
mgrpURLResultMapping: X-EXTLDAP-LISTS
```

with a mapping table

X-EXTLDAP-LISTS

```
extldap:///.*?listID?sub?*|*  $CROOT|$]extldap:///.$0?listMemberRoot?sub?$1[|$2
!
! Probe is now:
!  ROOT|listMemberRoot|listID
!
!  ROOT|*|*          $C$]extldap:///.$0?mail?sub?(isMember=$1)[ $E$Y
```

36.2.1.1.2 GROUP_TEMPLATES mapping table

The GROUP_TEMPLATES mapping table provides a way to support multiple ways of defining group membership: it extends the [group_dn_template MTA option](#) approach, allowing use of different "DN expansion templates" to combine with the values coming from the LDAP attributes named by the [ldap_group_dn](#) and [ldap_group_dn2](#) MTA options.

The LDAP attributes named by the [ldap_group_dn](#) and [ldap_group_dn2](#) MTA options are typically used to specify DN's, which are then expanded to find user entries using the URL template specified via the [group_dn_template MTA option](#). By setting a different sort of value for the [group_dn_template](#) MTA option, a different sort of DN expansion approach could be used -- but it would then apply to all values of the LDAP attributes named by both [ldap_group_dn](#) and [ldap_group_dn2](#). The GROUP_TEMPLATES mapping table, in contrast, can select alternate expansion approaches based on LDAP attribute name and value, thereby allowing support for multiple, different ways of expanding DN's to determine group membership.

When a GROUP_TEMPLATES mapping table exists, it will be probed each time a group has an LDAP attribute named by the [ldap_group_dn](#) or [ldap_group_dn2](#) MTA option to expand. The probe form is:

```
object-classes | attribute-name | attribute-value
```

where *object-classes* is a plus-separated list of the [object classes](#) associated with the current LDAP entry, *attribute-name* is the name of the group "DN" attribute being expanded (*i.e.*, the LDAP attribute name specified for either [ldap_group_dn](#) or [ldap_group_dn2](#)), and *attribute-value* is that attribute's current value.

If the mapping sets the `$Y` output flag, then the mapping output string will be used as the template for this attribute's expansion in place of using the value of `group_dn_template` as the template. If the mapping sets the `$N` output flag, then the attribute will be silently ignored.

36.2.2 Proper use of lists rather than groups

The fundamental difference between an e-mail group *vs.* a true mailing list is in how [notification messages](#) are defined to be handled: in particular, whether the original envelope From address (the error report address) is retained (a group) or overridden with a list owner address (a true mailing list). In the case of groups and mailing lists defined in LDAP, the distinction is whether an `mgrpErrorsTo` attribute has been set; in the case of groups and mailing lists defined via [alias options](#) in Unified Configuration, the distinction is whether the `alias_envelope_from` alias option has been set; in the case of groups and mailing lists defined via the [alias file](#), the distinction is whether the `[ENVELOPE_FROM] named parameter` or alternatively the `error-return-address` positional parameter has been set.

For almost all cases, true mailing lists should be used. Groups, on the other hand, should be reserved for those rare cases where:

- The group membership is (very!) small: on the order of a handful of members.
- The group membership is quite cohesive in terms of roles and relationships: examples would be multiple mailboxes for the same actual person, members of a (small!) family, emergency sysadmin contacts, *etc.*
- The group membership is quite static (unchanging).

Regrettably, (mis)use of simple groups when mailing lists would be more appropriate is wide spread--and survives simply because sending messages to a group is so easy and *until something goes wrong* the difference is not apparent. Of course, once something does go wrong, it's precisely the suboptimal handling of the notifications that becomes a problem for the (misused) groups.

Many (probably most) sites do not have this distinction clear at all; they think of a group (abstractly a set of users) as essentially synonymous with the use of that group as (the membership of) a mailing list. And it's routine to plaster a `mail` attribute onto every "group", at which point one *can* send messages to the group, making it a pseudo-mailing list. But just plastering a `mail` attribute onto a group does *not* make a group a mailing list, and in fact it is perhaps unfortunate that this is so "easy" as it lets sites make these sorts of conceptual and operational mistakes so easily. (There are two possible mistakes: the more common is to use a `mail-attribute-decorated group sans mgrpErrorsTo` as an autoforwarder in cases where it is not appropriate; but simple addition of `mgrpErrorsTo` to a `mail-attribute-decorated group` thereby turning it into a true list may be another, though lesser, sort of mistake, especially when [groups are nested](#), if one would prefer to still have the group available as a pure set of users, suitable as membership for some superset groups or lists). Except for certain special cases where it really does make sense to have a mail address for the group function as an autoforwarder, groups should not be used directly as a pseudo-list. When no nesting is needed, a group can be turned into a list definition by adding `mgrpErrorsTo`. Or most generally, for best overall practice, the definitions should be separate to correspond to the conceptual distinction: there is a group defining a related set of users, and then there is a separate list entry in LDAP that has that group as its membership.

So, for almost every case of an LDAP group entry to which messages will be addressed, correct definition should use an entry that includes an `mgrpErrorsTo` attribute (more precisely,

whatever attribute is pointed to by the `ldap_errors_to` MTA option), pointing to some "responsible" or "list owner" address, so that the group is also in e-mail terms, a true mailing list.

To expand upon the differences and options in notification handling:

- With a plain group definition, as of JES MS 6.1 syntax errors or other "immediately apparent" errors (such as over quota status for local users) in the member address definitions will be reported *to the entire rest of the group membership*, whereas with a mailing list definition such syntax errors/immediately apparent errors are reported only to the "responsible list owner" address: the `mgrpErrorsTo` address.

Note that in neither case is informing the original sender typically appropriate: the maintenance of the correctness/usability of the addresses in a group or list definition is not the business of the original message sender, but rather best handled by either the (close and cohesive) "fellow members" of the group in the case of a (properly used) group, or by the "responsible list owner" in the case of a true mailing list. The original message sender, in contrast, could well be some remote correspondent with no ability to "fix" the "bad" address definition, and in the case of (properly used) groups, where the membership of the group is quite cohesive and overlapping in intended roles so that as long as someone in the group got the message it can be considered successfully received, a bounce message back that an address was "bad" may be misleading and unnecessarily worrisome to the original sender. Note that such cases of immediately apparent as invalid addresses (but where at least one address in the group appears initially valid) would not get reported to the original message sender even prior to JES MS 6.1; instead such cases were silently ignored prior to JES MS 6.1. Note also that a case where all address definitions in a group are bad does, of course, get reported back to the original sender: in that case the sender's message was not received at all, and the entire group address was bad. The case discussed above is where at least *some* of the group addresses appeared initially valid.

- In the case of initially apparently valid addresses that suffer delivery problems, for a plain group such delivery problems are reported back to the original sender, whereas for a true mailing list the delivery problems are reported back to the "responsible list owner" address. With only a little thought, it should become clear why with any sort of large list, or any list where membership is at all subject to change (membership "turn over"), informing merely the list owner (who may want to consider removing chronically "bad" addresses from the list membership) is desirable, and why "informing" the original sender is likely to be perceived as "pestering" the original sender with irritating, useless, and irrelevant (to them) bounce messages. Furthermore, exposing who is on the list, but suffering delivery problems, to arbitrary original senders may be considered an undesirable exposure of information in some cases. (Keep in mind that some mailing lists and groups are configured so that being allowed to post to the list or group does not imply ability to see the list membership.) So notifying only the "responsible list owner" about delivery problems to list members is a true advantage that mailing lists have over groups. Now due to the widespread (mis)use of groups for mailing list purposes, users may have gotten accustomed, when posting to what is actually a group (but being misused as a mailing list), to receiving notifications of delivery problems to the individual group members. In such cases, the proper course is, almost always, to switch to a mailing list and educate the users on what to expect with a true mailing list, rather than continue to (mis)use a group.
- For the rare cases where mailing list semantics are appropriate in general, but where it truly makes sense to notify the original message sender of delivery problems to mailing list members, new in JES MS 6.3 (but not working until the fix for CR # 6530591), setting `mgrpErrorsTo: /`, that is, setting to the forward slash character, has a special meaning. It

tells the MTA to revert to using the original envelope From address that had been present on the incoming message, yet in all other respects use mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender. Note that this feature should *not* be used merely to avoid having to educate users about a change from group to mailing list semantics, as this is more than a "cosmetic" feature but rather has significant semantic implications; use of this feature should be reserved for cases where its semantic implications are truly understood and desired.

See the discussion of the `mgrpErrorsTo` attribute (or more precisely, the attribute named by the `ldap_errors_to` MTA option) for some additional discussion and special syntaxes for this attribute's value.

Another difference between groups *vs.* true mailing lists is in the area of passing through of NOTARY flags on the message "copies" to the various recipients; here too mailing list behavior tends to be distinctly more desirable, as the group behavior (as required by Internet standards!) is optimized for the case of multiple mailboxes (aliases) for a single person.

36.2.2.1 Nested groups and nested mailing lists

For many purposes, a useful thing to do is to "nest" group definitions: have groups defined that contain other subgroups as members, or which are themselves contained in bigger groups.

When wishing to use such sets of users as [mailing lists](#), however, usually a separate list definition should be made for *each* such desired list, defining each list's membership to be an appropriate group (or groups): for instance, using the `memberURL` attribute. For instance, schematically:

```
mail: list1@domain.com
mgrpErrorsTo: list1-owner@domain.com
memberURL: url-for-group2
memberURL: url-for-group3
```

where `group2` and `group3` are separately defined as *groups* with their own members, not as mailing lists (in particular, no `mgrpErrorsTo` attribute). Indeed, it may be desirable to define `group2` and `group3` so that they do not even have a `mail` attribute of their own (are not addressable as a pseudo-list). But if it is desired to also have the sets of users in `group2` and `group3` be accessible for e-mail postings, then use additional, separate list definitions, that in turn reference the `group2` and `group3` definitions for their membership, along the lines of:

```
mail: list2@domain.com
mgrpErrorsTo: list2-owner@domain.com
memberURL: url-for-group2
```

```
mail: list3@domain.com
mgrpErrorsTo: list3-owner@domain.com
memberURL: url-for-group3
```

where note that these are indeed new list definitions, that make use of separate and distinct group definitions for the membership.

That is, when nesting of groups (or so-called nesting of mailing lists) is neither used nor desired, then converting existing group definitions to true mailing lists definitions by simply

adding an `mgrpErrorsTo` attribute (and any other desired mailing list attributes) is perfectly fine. However, if nesting of groups (or mailing lists) is desired, then simply converting the group definitions to list definitions is not optimal. Instead, this is the case where the more general approach of having group definitions (possibly not even including a `mail` attribute for the group) and then separate list definitions that use the groups for membership, offers significant advantages, including:

- By having the list membership defined as combining multiple groups, or nesting groups, the LDAP search criteria on the membership uses a search filter that includes all the appropriate groups, so that duplicate elimination of members, from within all the groups and all the nested levels of groups, will happen when the MTA is initially expanding the membership of the list, rather than expanding to separate sub-lists that then do their own membership expansion operation limited to their own membership. That is, this approach allows better elimination of duplicate addresses--a list member who is a member of multiple subgroups defining the list will get one message copy.¹
- By not using nested lists, list-specific modifications to messages, such as additions of List-*: header lines, setting of the `error return address` (the list "owner" address), *etc.*, will be performed once, consistently for the entire list membership, rather than having nested levels of changes occurring at each stage of nesting, causing changes to be cumulative or override each other.

¹ A separate issue is the so-called "duplicate" message copy -- which is not truly an exact duplicate -- that a recipient will receive if the original message was addressed *directly* to them, *e.g.*, on the To: or Cc: header line, as well as being addressed to the list of which the recipient is also a member. A message copy resulting from a posting to a list is fundamentally different from the copy addressed directly to the recipient: even though the message content may be superficially "the same" (though even that is not necessarily the case, if list-specific text addition, or spam/virus cleaning, or document conversion, or character set translation, or other modifying operations have occurred), the notification address will be different for a true mailing list copy. Other differences, such as in NOTARY flags, or in Received: header lines, *etc.*, also typically exist between mailing list copies and "direct" copies. Any difference that requires different handling by MTAs handling the message means that a different copy must be created at that point, and once a message has bifurcated in such a way, further divergence is possible and even likely: separate copies *will* be delivered and they are only superficially "duplicates". So keep in mind that even if the copies do not seem "significantly" different to the recipient, some difference existed at some point in time. Furthermore, sophisticated users may themselves wish to perform different handling of different types of incoming messages; for instance, some users will subscribe to mailing lists using subaddresses, thereby allowing for more [convenient Sieve filtering](#) of their incoming list messages.

36.2.3 Restricting posting access to large lists

Especially with a relatively "large" mailing list, it is usually wise to enforce at least some restrictions on who is allowed to post (send) to the list, so that the list is not used as an easy mechanism by which to spam the members. The MTA supports a variety of forms and mechanisms for such restrictions. For "large" mailing lists, more secure forms of restriction such as password-protected list access, or posting restricted to explicitly listed senders who are required to authenticate (use SMTP AUTH) themselves when submitting, may be especially appropriate. (Note that setting such posting access controls also limits who is allowed to view the membership of the list via the SMTP EXPN command---which may be beneficial in limiting address harvesting by spambots.)

With large mailing lists, setting

```
mailDeferProcessing: AFTER_AUTH
```

(which setting is only available and valid in JES MS 6.3p1 and later) is especially desirable. [This setting](#) causes immediate checks of any access controls, but deferred expansion of the list membership. This then allows immediate rejection of messages that do not meet posting criteria, while deferring the (possibly time consuming) list membership expansion until later, off-line, when the [reprocess channel](#) runs.

For instance, to permit postings only when the sender authenticated (using their account password) as either `mailadmin1@domain.com` or `mailadmin2@domain.com`:

```
mgrpBroadcasterPolicy: SMTP_AUTH_REQUIRED
mgrpAllowedBroadcaster: mailadmin1@domain.com
mgrpAllowedBroadcaster: mailadmin2@domain.com
```

Or to permit postings only when the sender provided a secret password on an Approved: header line (which same header line the MTA will automatically remove from the message distributed to list members):

```
mgrpBroadcasterPolicy: PASSWORD_REQUIRED
mgrpAuthPassword: secret-password
```

For many lists, an appropriate, less stringent restriction is to limit postings to members of the lists. The check on posters may be based simply on the attempting poster's e-mail address; for instance:

```
mgrpAllowedBroadcaster: mailto:list-address
```

or may further require that a poster in fact authenticated as a list member:

```
mgrpBroadcasterPolicy: SMTP_AUTH_REQUIRED
mgrpAllowedBroadcaster: mailto:list-address
```

Note that requiring SMTP AUTH use for postings usually also implicitly requires that all members of the list be "local" members (have a local account/be able to authenticate). (Though by trusting passed-along authentication from other systems, or by combining sub-list definitions appropriately, it is possible to achieve an effect whereby "local" users must authenticate to post, while still allowing postings from external users who are not capable of authenticating against your user directory.)

Or yet another routinely useful sort of list posting restriction is to allow direct postings only by members of the list, while redirecting any attempted postings by non-members to a [list moderator](#); for instance:

```
mail: list-y@domain.com
mgrpMsgRejectAction: toModerator
mgrpAllowedBroadcaster: mailto:list-y@domain.com
```

```
mgrpModerator: mailto:list-y-owner@domain.com  
mgrpErrorsTo: list-y-owner@domain.com
```

For additional flexibility in posting access controls, see the [GROUP_AUTH mapping table](#).

36.2.3.1 GROUP_AUTH mapping table

The MTA's [group/list access control mechanisms](#) allow for a wide variety of access and permission models. However, exploiting this flexibility often depends on being able to define what attributes and values appear in LDAP group entries. If the entries being processed cannot be modified, as for instance in the case of an [externally controlled LDAP directory](#), it becomes necessary for the MTA to adopt a more flexible processing model in order to support different attribute syntaxes.

New in 7.0.5, the `GROUP_AUTH` mapping table and four new MTA options `ldap_auth_mappingN` ($N=1-4$) have been added to facilitate such processing. The MTA options are used to specify the names of up to four additional LDAP attributes to be fetched during alias expansion processing. When the `GROUP_AUTH` mapping is defined and at least one of the four attributes `ldap_auth_mappingN` is defined and appears on a group, then the `GROUP_AUTH` mapping is probed during group authorization checks (before any other authorization checks are done). The probe format is:

```
envelope-from | group-address | auth1 | auth2 | auth3 | auth4
```

Here the `authN` fields are simply whatever values are associated with the `ldap_auth_mappingN` named LDAP attributes for this group. If multiple attributes or multiple attribute values appear, they will all be present in the probe field, separated by commas.

The `GROUP_AUTH` mapping can produce any of four possible outputs:

- `$Y` indicates that the authorization check has passed.
- `$T` indicates that the mapping result is a URL, which is then checked in the same fashion as an `ldap_auth_url` would be.
- `$N` indicates that authorization has failed.
- `$F` indicates that the mapping result is a URL, which is then checked in the same fashion as an `ldap_cant_url` would be.

36.2.4 Performance submitting mass mail messages

Deferred processing of the expansion of the list membership, with or without deferred processing of list posting access controls, and setting of relevant bits of the `ldap_use_async` MTA option, tends to be especially important to consider for [truly large lists](#).

For controlling and enabling the deferred processing of expansion of membership and/or access controls, see especially the per-list `mailDeferProcessing` LDAP attribute (more precisely, the LDAP attribute named by the `ldap_reprocess` MTA option), as well as the default set via the `defer_group_processing` MTA option. For large lists, it is usually desirable to set either the (new in JES MS 6.3p1) value of `-1` for `defer_group_processing`, or equivalently the per-list setting of `AFTER_AUTH` for the `mailDeferProcessing` attribute; this allows inline processing of access controls, while deferring membership expansion. Alternatively, when the access control itself consists of a very large list, it

may be desirable to defer the access control processing as well, as selected via a value of `defer_group_processing=1` or the per-list setting of Yes for the `mailDeferProcessing` attribute.

36.3 Special address formats for list members

There are a few special address formats that may be worth considering using for list members, or that users may wish to use if subscribing themselves to a mailing list.

For one thing, it may be worth considering subscribing using a [subaddress](#), to aid in convenient processing, *e.g.*, by a [Sieve filter](#), of the mailing list postings when they are received. Such Sieve filter processing might include automatically [filing mailing list postings into a special folder](#) based on the [subaddress](#) present on the recipient address, or specialized processing if one is the list moderator; see examples in [Moderated mailing lists](#).

On a list member address, for MTA mailing lists defined via [alias options](#), the [alias file](#) or [alias database](#), a comment string containing the special string (by default "NOPOST") defined by the `post_off` MTA option disables the ability for the list member to post: the list member can receive list messages, but may not post. Using this comment string on every list member would be one way (though not necessarily the simplest way -- see [Restricting posting access to large lists](#)) to set up a broadcast-only (no posting) mailing list.

On a list member address, for MTA mailing lists defined via [alias options](#), the [alias file](#) or [alias database](#), a comment string containing the special string (by default "NOMAIL") defined by the `mail_off` MTA option disables the ability for the list member to receive list postings: the list member might be able to make list postings (depending upon any relevant [mailing list access controls](#)), but will not receive list postings. In particular, a [list moderator](#) may wish to use this feature -- the list moderator will see the list messages initially during the moderation phase, and may not wish to receive the messages again when the actual posting goes through.

For aliases defined via user LDAP entries, note that as of MS 7.0.5 a `mailDeliveryOption` (or whatever LDAP attribute is named by the `ldap_delivery_option` MTA option) value of `nomail` is considered valid, but will not receive any messages (messages are discarded). This is not suitable for the normal moderator use mentioned above as such a user entry receives *no* mail, but it is suitable for unmonitored mailboxes, when it is not intended that the mailbox should ever receive *any* messages. And it may be suitable for cases where mail forwarding or application of a user Sieve filter is desired (but normal receipt of messages is not desired).

(New in MS 8.0.1.) On a list member address, a comment string containing the special string (by default "ALTERNATE-RECIPIENT") defined by the `alternate_recipient` MTA option will cause the address specified in that comment string to be used as an alternate delivery address, for messages which cannot be delivered to this primary list member address.

Chapter 37 Mapping tables

37.1 Mapping table format	37-2
37.1.1 Mapping table format in legacy configuration	37-2
37.1.2 Mapping table format in Unified Configuration	37-3
37.1.3 Mapping entry patterns	37-4
37.1.4 Mapping entry templates	37-7
37.2 The mapping group	37-17
37.3 Mapping operation	37-18
37.4 Handling large numbers of mapping table entries	37-18
37.4.1 General database	37-20
37.5 When mapping table changes take effect	37-21
37.6 Pre-defined mapping tables	37-21
37.7 Testing mapping tables	37-22
37.7.1 Testing address access mapping tables	37-23
37.8 Callout routines	37-24
37.8.1 check_memcache.so callout	37-24
37.8.2 check_metermaid callouts	37-27
37.8.3 dns_verify callouts	37-28

Many components of the MTA employ table lookup oriented information. One particular type of table is used more often in the MTA than any other. Generally speaking, this sort of table is used to transform (*i.e.*, map) an input string into an output string. Such tables, referred to as 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. Indeed, most of the [MTA crdbdatabases](#) can be considered instances of just this sort of mapping table. MTA crdb database files, however, do not provide wildcard lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

For more flexible, pattern-based mappings, the MTA also supports its own mapping tables. In legacy configuration mode, such MTA mapping tables are stored in the MTA mapping file; in Unified Configuration, MTA mapping tables are [mapping](#) XML elements and they may be referenced from within the `msconfig` utility as

```
mapping:mapping-name
```

where such a mapping table named `mapping-name` consists of an ordered list of rules. 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.

A fairly complete list of the mapping table names always recognized by the MTA is available under [Pre-defined mapping tables](#). Some of the most commonly used mapping tables are the [access mapping tables](#) used to control who can send and receive e-mail. Sites may also define arbitrary mapping tables.

You can test general mapping table processing with the `imsimta test -mapping` utility. (Note that the `imsimta test -mapping` utility tests only the general-to-all-mapping-

tables functionality. It is not specific for testing of the functional meaning of specific mapping tables, such as access controls due to [address-based *_ACCESS mapping tables](#), or address changes due to the [REVERSE mapping table](#). Instead, the `imsimta test -rewrite` utility is typically more useful for such functional or semantic testing.)

37.1 Mapping table format

Mapping tables in legacy configuration were stored in the MTA mappings file; see [Mapping table format in legacy configuration](#) for further discussion.

In Unified Configuration, mapping tables are stored as part of the Unified Configuration. They may be viewed and modified in an editor "as if" they were still in the old mappings file format, or they may be viewed and modified as Unified Configuration rules; see [Mapping table format in Unified Configuration](#) for further discussion.

37.1.1 Mapping table format in legacy configuration

The mapping 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. Prior to 7.0.5, any spaces within an entry must be quoted with the dollar sign, `$`. It is required that a blank line appear after each mapping table name and between each mapping table; no blank lines may appear between entries in a single table. Comments are introduced by an exclamation mark, `!`, appearing in the first column.

Pictorially, the format that results looks like this:

TABLE-1-NAME

```

    pattern1-1    template1-1
    pattern1-2    template1-2
    pattern1-3    template1-3
    .
    .
    .
    pattern1-n    template1-n

```

TABLE-2-NAME

```

    pattern2-1    template2-1
    pattern2-2    template2-2
    pattern2-3    template2-3
    .
    .
    .
    pattern2-n    template2-n

```

```

    .
    .
    .

```

```
TABLE-m-NAME
```

```

.
.
.

```

In this example an application using the mapping table `TABLE-2-NAME` would map the string `pattern2-2` into whatever is specified by `template2-2`.

A mapping table name may be up to 128 characters long. Each pattern and each template can contain up to 256 characters before substitutions; the result of applying substitutions must be no more than 1024 characters each. As of JES MS 6.3p1, the template (right hand side) limit has been increased to 1024 characters, and the overall length of each line in the mapping table has been increased to 4096 characters. (Prior to JES MS 6.0, the mapping table name was limited to 64 characters while the pattern and template were each individually limited to at most 252 characters.) There is no limit to the number of entries that can appear in a mapping (although excessive numbers of entries may eat up huge amounts of CPU and can consume excessive amounts of memory). Long lines 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.

When the MTA probes a mapping table, the overall probe length is also limited to 1024 characters.

Duplicate mapping table names are not allowed in the mapping file.

37.1.1.1 Including other files in the mapping file

Other files may be included in the mapping file. This is done with a line of the form:

```
<file-spec
```

This will effectively substitute the contents of the file `file-spec` into the mapping file at the point where the include appears. The file specification should specify a full file path (device, directory, *etc.*). All files included in this fashion must be world readable. Comments are also allowed in such included mapping files. Includes can be nested up to three levels deep. Include files are loaded at the same time the mapping file is loaded --- they are not loaded on demand, so there is no performance or memory savings involved in using include files.

37.1.2 Mapping table format in Unified Configuration

In Unified Configuration, mapping tables may be edited from within `msconfig` using a command such as

```
msconfig> edit mappings mapping-name
```

which will present the mapping table in the familiar [tabular form of legacy configuration](#).

Alternatively, the XML mapping elements may also be referenced from within `msconfig` as an ordered list of rule settings grouped in a [mapping](#), *e.g.*,

```
mapping:ORIG_SEND_ACCESS.rule = tcp_local|*|tcp_local|* $N$D30|Relaying not allowed
mapping:ORIG_SEND_ACCESS.rule = tcp_*|*|native|* $N
```

```

mapping:ORIG_SEND_ACCESS.rule = tcp_*|*|hold|* $N
mapping:ORIG_SEND_ACCESS.rule = tcp_*|*|pipe|* $N
mapping:ORIG_SEND_ACCESS.rule = tcp_*|*|ims-ms|* $N
mapping:ORIG_SEND_ACCESS.rule = tcp_local|*|tcp_intranet|@*:*.* $N$D30|Explicit routing not allowed
mapping:ORIG_SEND_ACCESS.rule = tcp_local|*|tcp_intranet|*%*.* $N$D30|Explicit routing not allowed
mapping:ORIG_SEND_ACCESS.rule = tcp_local|*|tcp_intranet|*!*.* $N$D30|Explicit routing not allowed
mapping:ORIG_SEND_ACCESS.rule = tcp_local|*|tcp_intranet|"*@*.* $N$D30|Explicit routing not allowed
    
```

37.1.3 Mapping entry patterns

Mapping patterns can contain wildcard characters. In particular, the usual so-called "glob style" wildcard characters are allowed: an asterisk, *, will match zero or more characters and each percent sign, %, will match 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. Additional wildcards available in patterns are listed in the table shown below.

Table 37.1 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 <i>n</i> th wildcard or glob
Modifiers	Description
$\$_\dagger$	Use minimal or "lazy" left-to-right matching
$\$@\dagger$	Turn off "saving" of the succeeding wildcard or glob
$\†	Turn on "saving" of the succeeding wildcard or glob; this is the default
Glob wildcard	Description
$\$A$ or $\$A\%$	Match one alphabetic character, A-Z or a-z
$\$A^*$	Match zero or more alphabetic characters, A-Z or a-z
$\$B$ or $\$B\%$	Match one binary digit (0 or 1)
$\$B^*$	Match zero or more binary digits (0 or 1)
$\$C$ or $\$C\%$	(New in 7.0) Match one ASCII control character (other than horizontal TAB)
$\$C^*$	(New in 7.0) Match zero or more ASCII control characters (other than horizontal TAB)
$\$D$ or $\$D\%$	Match one decimal digit 0--9
$\$D^*$	Match zero or more decimal digits 0--9
$\$H$ or $\$H\%$	Match one hexadecimal digit 0-9 or A-F
$\$H^*$	Match zero or more hexadecimal digits 0-9 or A-F
$\$O$ or $\$O\%$	Match one octal digit 0-7
$\$O^*$	Match zero or more octal digits 0-7

<code>\$S</code> or <code>\$S%</code>	Match one symbol set character, <i>i.e.</i> , 0-9, A-Z, a-z, <code>_</code> , <code>\$</code>
<code>\$S*</code>	Match zero or more symbol set characters, <i>i.e.</i> , 0-9, A-Z, a-z, <code>_</code> , <code>\$</code>
<code>\$T</code> or <code>\$T%</code>	Match one tab or vertical tab or space character
<code>\$T*</code>	Match zero or more tab or vertical tab or space characters
<code>\$X</code> or <code>\$X%</code>	A synonym for <code>\$H%</code>
<code>\$X*</code>	A synonym for <code>\$H*</code>
<code>[\$c]</code> or <code>[\$c]%</code>	Match character <i>c</i>
<code>[\$c]*</code>	Match zero or more occurrences of character <i>c</i>
<code>[\$c₁c₂...c_n]</code>	Match exactly one occurrence of character <i>c</i> ₁ , <i>c</i> ₂ , or <i>c</i> _n
<code>[\$c₁c₂...c_n]%</code>	Match exactly one occurrence of character <i>c</i> ₁ , <i>c</i> ₂ , or <i>c</i> _n
<code>[\$c₁c₂...c_n]*</code>	Match zero or more occurrences of any characters <i>c</i> ₁ , <i>c</i> ₂ , or <i>c</i> _n
<code>[\$c₁-c_n]</code>	Match any one character in the range <i>c</i> ₁ to <i>c</i> _n
<code>[\$c₁-c_n]%</code>	Match any one character in the range <i>c</i> ₁ to <i>c</i> _n
<code>[\$c₁-c_n]*</code>	Match zero or more occurrences of characters in the range <i>c</i> ₁ to <i>c</i> _n
<code>\$<IPv4></code>	Match an IPv4 address, ignoring bits
<code>\$(IPv4)</code>	Match an IPv4 address, keeping prefix bits
<code>\${IPv6}</code>	Match an IPv6 address

(‡) When combined with a back match or glob wildcard, only one dollar sign character is used for both; for instance, `$_0*` is a non-saved back match of the first (0th) wildcard, and similarly, `$_@A` is a non-saved alphabetic character glob wildcard.

Within globs, *i.e.*, 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.

Note that to specify multiple modifiers, or to specify modifiers and a back match, or to specify modifiers and a glob, 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 `$_0*`, *not* `$_@$_0*`. Similarly, to match a decimal digit without saving the matched digit, one would use `$_@D`, *not* `$_@$_D`.

Note that the `imsimta test -match` utility may be used to test mapping patterns and specifically to test wildcard behavior in patterns.

37.1.3.1 Back matching with `$n*`

In some mapping tables, it is particularly useful to be able to compare whether two portions (fields) of the mapping table input are identical. The "back match" wildcards are provided for this purpose. Back matches of `$_0*` through `$_9*` are available. For instance, a `$_0*` wild card looks for (matches on) the same characters already matched in the very first wildcard or glob in an entry, while a `$_1*` wildcard looks for the same characters already matched in the second wildcard or glob in an entry, *etc.*

With IPv4 "prefix bits" 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 instance,

```
$(123.45.67.0/24)
```

will match anything in the 123.45.67.0 class C subnet.

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 instance,

```
$<123.45.67.0/8>
```

will match anything in the 123.45.67.0 subnet. Or another example is that

```
$<123.45.67.4/2>
```

will match anything in the range 123.45.67.4--123.45.67.7.

IPv6 matching matches an IPv6 address or subnet. The syntax is:

```
${xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx/n}
```

where each `xxxx` can consist of one to four hexadecimal digits (digits in the range 0 to F) and where `n` is an integer in the range 0 to 128 specifying the number of prefix bits that must match. In particular, each additional 16 bits of prefix matching requires that another 4 hexadecimal digit "chunk" of the IPv6 address must match. For instance,

```
${12AF:0:0:0:0:0:0:0/16}
```

means to match any IPv6 address where the first sixteen bits correspond to the hexadecimal value 12AF, regardless of what the remaining 112 bits in the address may be.

For IPv6 `${}` matching, chunks may be omitted using the `::` marker. For instance, `${12AF:0:0:0:0:0:0:0/16}`, `${12AF::/16}`, and `${12AF::0/16}` are all equivalent. In particular, a commonly used example is that to match the local host, one may use

```
${::1}
```

meaning to match `0:0:0:0:0:0:0:1`.

See the [INTERNAL_IP mapping table](#) for examples of IP address matching.

37.1.4 Mapping entry templates

If the comparison of the pattern in a given mapping entry fails, no action is taken; the scan proceeds to the next entry. If the comparison succeeds, the right hand 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 will not appear in the output string produced by a template. See the table below for a list of the special substitution and standard processing metacharacters. Any other metacharacters are reserved for mapping-specific applications.

Note that any of the metacharacters \$C, \$E, \$L, or \$R, when present in the template of a matching pattern, will influence the mapping process, controlling 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 again iterate a mapping is not honored after the counter has exceeded 10.

Table 37.2 Mapping template substitutions and metacharacters

Substitution sequence	Substitutes
$\$n$	<i>n</i> th wildcarded field as counted from left to right starting from 0
$\$\%...\%$	(New in JES MS 6.2.) Substitute in a load average value.
$\#\dots\#$	Sequence number substitution.
$\$+n\#\dots\#$	Hash substitution.
$\$\&...\&$	(New in JES MS 6.2.) Character value substitution.
$\$]...\[$	LDAP search URL lookup; substitute in result (first value, if multiple values returned).
$\$]\$]...\[$	LDAP search URL lookup requiring that only one value be returned; substitute in result.
$\$.text.$	(New in JES MS 6.3.) Result to use in the case of LDAP lookup temporary failures.
$\$}\dots\{$	LDAP domain map attribute lookup; substitute in attribute value.
$\$\{\dots\}$	General database substitution.
$\$\ \dots\ $	Apply specified mapping table to supplied string.
$\$\`...\`$	Evaluate expression; substitute in result.
$\$\[...\]$	Invoke site-supplied routine; substitute in result.
Metacharacter	Description

\$C	Continue the mapping process starting with the next table entry; use the output string of this entry as the new input string for the mapping process.
\$E	End the mapping process now; use the output string from this entry as the final result of the mapping process.
+\$1E	(New in JES MS 6.3.) End the mapping process now and (unlike \$E) do not even interpret the rest of the template; use the output string from this entry (sans interpretation of the rest of the template) as the final result of the mapping process.
\$L	Continue 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 make one additional pass starting with the first table entry. A subsequent match may override this condition with a \$C, \$E, or \$R metacharacter.
\$R	Continue the mapping process starting with the first entry of the the mapping table; use the output string of this entry as the new input string for the mapping process.
\$?x?	Mapping entry succeeds <i>x</i> percent of the time.
\$\	Force subsequent text to lowercase.
^	Force subsequent text to uppercase.
\$_	Leave subsequent text in its original case (and without LDAP URL style quoting).
\$=	Force subsequent material to be properly quoted (encoded) according to LDAP URL syntax rules.
:\$x	Match only if the specified flag is set.
;\$x	Match only if the specified flag is clear.

37.1.4.1 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 entry

```
PSI$%*::*$1@$0.psi.network.org
```

would match the input string PSI%A::B and produce the resultant output string b@a.psi.network.org. The input string PSI%1234::USER would also match producing USER@1234.psi.network.org as the output string. The input string PSIABC::DEF would not match the pattern in this entry and no action would be taken; *i.e.*, no output string would result from this entry.

37.1.4.2 Controlling text case, \$\, \$^, \$_

\$\$ forces subsequent text to lowercase, \$^ forces subsequent text to uppercase, and \$_ causes subsequent text to retain its original case (and turns off LDAP URL quoting if it had been previously turned on via the \$= metacharacter. For instance, these metacharacters may be useful when using mappings to transform addresses for which case is significant.

37.1.4.3 Processing control, \$C, \$L, \$R, \$E, \$+1E

The `$C`, `$L`, `$R`, and `$E` metacharacters influence the mapping process, controlling whether and when the mapping process terminates. `$C` causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process. `$L` causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process, and, if no matching entry is found, making one additional pass through the table starting with the first table entry; a subsequent matching entry with a `$C`, `$E`, or `$R` metacharacter overrides this condition. `$R` causes the mapping process to continue from the first entry of the table, using the output string of the current entry as the new input string for the mapping process. `$E` causes the mapping process to terminate; the output string of this entry is the final output. `$E` is the default.

New in JES MS 6.3 is the `+$1E` metacharacter. It acts like `$E`, except that unlike `$E` it inhibits (stops) interpretation of the rest of the template.

Mapping table templates are scanned left to right. So to set a `$C`, `$L`, or `$R` flag for entries that may "succeed" or "fail", *e.g.*, general database substitutions, or random value controlled entries, put the `$C`, `$L`, or `$R` metacharacter to the left of the part of the entry that may succeed or fail; otherwise, if the remainder of the entry fails, the flag will not be seen.

37.1.4.4 Check for special flags

Some mapping probes have special flags set. `$.x` causes an entry to match only if the flag `x` is set. `;$x` causes an entry to match only if the flag `x` is clear. See specific mapping table descriptions 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` metacharacter to the right of the flag check.

37.1.4.5 Entry randomly succeeds or fails, `$?x?`

`$?x?` in a mapping table entry causes 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 argument between the `?`'s, `x`, should consist of 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 e-mail and you'd like to slow it down; if you're using the MTA's SMTP server on port 25, 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 temporarily 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 a temporary SMTP error (in this example, 452 4.4.0) and the text message "Try again later".

```
PORT_ACCESS
```

```
TCP | * | 25 | 123.45.6.78 | *      $C$?25?$Y
```

```
TCP|*|25|123.45.6.78|*      $N452$ 4.4.0$ Try$ again$ later
```

Another example would be randomly issuing a temporary failure message for a certain percentage of SMTP messages from a particular envelope From: address; for instance, suppose the goal is to issue a temporary failure message with extended SMTP code 4.5.9 to 80 percent of the messages that busybee@some.where attempts to send to your local channel users. Then a SEND_ACCESS mapping table could be used, e.g.,

```
SEND_ACCESS
```

```
tcp_*|busybee@some.where|1|*      $C$?20?$Y
tcp_*|busybee@some.where|1|*      $N$X4.5.9|Try$ again$ later
```

37.1.4.6 Load average substitutions, #%...%

At the present time load average substitutions are not implemented.

37.1.4.7 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 unifier in the mapping table output; for instance, when using a mapping table to generate file names.

Permitted syntax is any one of:

```
 $#seq-file-spec#
```

or

```
 $#seq-file-spec|radix#
```

or

```
 $#seq-file-spec|radix|width#
```

or (new in JES MS 6.1)

```
 $#seq-file-spec|radix|width|modulus#
```

where the optional `seq-file-spec` argument is a full file specification for an (already existing) MTA sequence file, and where the optional `radix`, `width`, and `modulus` arguments specify the radix (base) in which to output the sequence value, the number of digits to output, and the modulus, respectively. The `seq-file-spec` argument may be omitted, in which case the MTA will use its own temporary sequence file (that will be created and used for the duration of this image). 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 will be padded with 0's on the left to obtain the desired number of digits. Note that if a width is explicitly specified, then the radix must be explicitly specified

also. If a modulus is specified, then the value inserted is the sequence number retrieved from the file mod the modulus; the default (and the only behavior available prior to JES MS 6.1) is not to perform any modulus operation.

As noted above, when specifying an explicit sequence file in a mapping, that file must already exist. To create a proper sequence file, on UNIX use the command:

```
% touch seq-file-spec
```

or

```
% cat >seq-file-spec
```

or on OpenVMS use the command

```
$ CREATE/FDL=PMDF_COM:sequence_number.fdl seq-file-spec
```

or on NT use the command

```
C:\> copy nul seq-file-spec
```

A sequence number file accessed via a mapping table must be world readable in order to operate properly.

37.1.4.8 Hash substitutions, $\$+n\#\dots\#$

(New in 7.4-18.01.)

A $\$+n\#\dots\#$ substitution, $n > 0$, computes a hash of a specified string and inserts the hash result into the mapping result. The value n selects the type of hash to compute.

Permitted syntax is any one of:

```
 $\$+n\#\text{string}\#$ 
```

or

```
 $\$+n\#\text{string}|\text{radix}\#$ 
```

or

```
 $\$=N\#\text{string}|\text{radix}|\text{width}\#$ 
```

```
 $\$+N\#\text{string}|\text{radix}|\text{width}|\text{modulus}\#$ 
```

where the *string* argument is the string to be hashed, and where the optional *radix*, *width*, and *modulus* arguments specify the radix (base) in which to output the hash value, the number of digits to output, and the modulus, 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 will be padded with 0's on the left to obtain the desired number of digits. Note that if a width is explicitly specified, then the radix must be explicitly specified also. If a modulus is specified, then the value inserted is the sequence number retrieved from the file mod the modulus; the default is not to perform any modulus operation on the hash value.

As noted above, the value of *n* selects the hash operation to perform. Currently the only supported value for *n* is 1, which employs the following hash function (32 bit integers are assumed):

```
int hashvalue(char *string, int length)
{
    unsigned int hash;
    int i, j;
    unsigned int *uptr;

    uptr = (unsigned int *)string;
    hash = length;
    j = length >> 2;
    for (i = 0; i < j; i++)
    {
        hash ^= *uptr++;
        hash = (hash << 9) | (hash >> 23);
    }
    for (i = j << 2; i < length; i++)
    {
        hash ^= string[i];
        hash = (hash << 13) | (hash >> 19);
    }
    return ((hash * 0x71279461U) >> 2);
}
```

37.1.4.9 Character value substitutions, \$&...&

(New in JES MS 6.2.) The MTA can generate UTF-8 strings from Unicode character values using the \$&...& substitution sequence. Multiple Unicode character values may be specified, by separating them with the comma character. For instance, substitution sequence of the form:

```
$&A0A0,20,A1A1&
```

will produce a UTF-8 string containing the characters at position A0A0, 20, and A1A1.

37.1.4.10 LDAP query URL substitutions, \$]...[, \$]\$,]...[, \$=, \$.

A substitution of the form \$]ldap-url[or \$]\$]ldap-url[is handled specially. ldap-url is interpreted as an LDAP query URL and the result of the LDAP query is substituted. The difference between the two forms is that in the case of a multi-valued result, \$]ldap-url[uses the "first" returned value (note that the LDAP protocol leaves the order of values returned as unspecified and implementation-dependent -- thus you must not make any assumptions about which value might be returned "first"), whereas a multi-valued response is an error (the lookup is considered to have failed) for \$]\$]ldap-url[.

Standard LDAP URLs as per [RFC 2255](#) are used, with the host and port typically omitted. (In versions prior to 7.4-18.01, the host and port had to be omitted; as of version 7.4-18.01, specifying the LDAP host and port in the URL is permissible.) If the host and port are omitted, they are assumed from the values of the `ldap_host` and `ldap_port` MTA options (or the `ugldaphost` and `ugldapport` base options, if the MTA options are not set, corresponding to the legacy `configutil` parameters (or `local.ugldaphost` and `local.ugldapport`). 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 what 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.

As of Messaging Server 7.2-7.02, so-called [external LDAP \(extldap:\) URLs](#) are also supported. (As of the Messaging Server 7.0u4 support for explicit host and port specification directly in the LDAP URL, this functionality is somewhat redundant. But it may still be useful when a particular external directory, and perhaps one with different authentication credentials needed for access, is used.) That is, if a URL is specified as

```
extldap:///dn[?attributes[?scope?filter]]
```

then an LDAP lookup will be performed against the alternate LDAP directory configured (via the [LDAP external directory lookup MTA options](#)) as the "external LDAP" directory.

Note that LDAP URLs have special character quoting (encoding) requirements. (See [RFC 1738, Section 2.2, "URL Character Encoding Issues"](#), and [RFC 2254, Section 4, "String Search Filter Definition"](#). Note that the quoting rules in [RFC 1960, Section 3, "String Search Filter Definition"](#), have been superseded by those of [RFC 2254](#).) The `$=` metacharacter forces subsequent material to be properly quoted (encoded) for LDAP URL usage; that is, any of the characters

```
$ & + , : ; = ?
```

will be converted to the percent character, "%", followed by the hexadecimal representation of their location in US-ASCII, any of the characters

```
( ) *
```

will be converted to "%5C" followed by the hexadecimal representation of their location in US-ASCII (the encoded form of the backslash character followed by the hexadecimal for the particular character), while the backslash character itself

```
\
```

will be converted to "%5C5C".

The `$_metacharacter` disables such LDAP quoting.

So note that when using a `$] . . . [` LDAP URL callout, one should normally use the `$=` and `$_` metacharacters around and substituted material that might contain special characters, and manually encode any fixed special characters in the material in the callout, or use just the `$=` and `$_` metacharacters around the entire interior of the LDAP URL body, *e.g.*,

```
$]ldap:///$=...$_[
```

The overall length of the LDAP URL (after any substitutions are performed) is limited to 252 characters in iMS 5.2, limited to 256 characters in JES MS 6.0 through JES MS 6.2, and limited to 1024 characters as of JES MS 6.3. Note also that the length of the original template in which such an LDAP URL appears is limited: to 252 characters in iMS 5.2 and earlier, or to 256 characters as of JES MS 6.0 and later; but substitutions in the template, and in particular substitutions used to construct the LDAP URL, may increase the LDAP URL length.

(New in JES MS 6.3.) The `$.text.` sequence can be used in a mapping entry to establish a string `text` which will be processed as the mapping entry result in the event of a temporary LDAP lookup failure. By default temporary LDAP failures cause the current mapping entry to fail, which is problematic in cases where different actions need to be taken depending on whether the LDAP lookup failed versus the directory server being unavailable or misconfigured. Once a failure string has been set using this construct, it will remain set until current mapping processing is completed. `$. .` can be used to return to the default state where no temporary failure string is set and temporary LDAP failures cause mapping entry 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.

37.1.4.11 LDAP domain map attribute substitutions, `$}...{`

(New in JES MS 6.1p1/JES MS 6.2.) A substitution of the form `$}domain-name,attribute{` is handled specially. `domain-name` is looked up in the directory as a domain, with domain map processing of the `domain-name` to build a proper LDAP query URL being automatically performed by the MTA. (In particular, in Schema 1 mode, the `domain-name` is looked up in the DC portion of the directory; while in Schema 2 mode, the `domain-name` is looked up using the Schema 2 domain filter.) Note that this argument `domain-name` truly is a domain *name*. If the domain `domain-name` exists and has the specified `attribute`, then the attribute's initial value is substituted into the mapping result; if either the attribute or the domain does not exist, then the mapping entry fails.

The following special syntaxes are also supported:

Table 37.3 Domain map mapping template special syntaxes

Syntax	Interpretation and effect
<code>\$}domain,_base_dn_{</code>	Return the base DN for the user/group entries belonging to this domain
<code>\$}domain,_domain_dn_{</code>	Return the DN of the domain entry itself
<code>\$}domain,_domain_name_{</code>	Return the name of the domain (as opposed to an alias)
<code>\$}domain,_canonical_name_{</code>	Return the canonical name associated with the domain

(New in 8.0.) A substitution of the form `$}user-identifier,attribute{` is handled specially. *user-identifier* is looked up in the directory as a user name.

The following attributes can be specified:

Table 37.4 User identifier mapping template attributes

Syntax	Interpretation and effect
<code>\$}user-identifier,#canonical_user#{</code>	Return the canonical form of user identifier
<code>\$}user-identifier,#canonical_domain#{</code>	Return the canonical domain the user is associated with
<code>\$}user-identifier, _#user_dn#{</code>	Return the DN of the user's entry in the directory

37.1.4.12 General database substitutions, `${...}`

A substitution of the form `$(text)` is handled specially. The *text* part is used as a key to access the MTA's [general database](#). If *text* is found in the database, the corresponding template from the database is substituted. If *text* does not match an entry in the database, the input string is used unchanged as the output string.

Depending upon the setting of the [use_text_databases](#) MTA option, the general "database" is either stored and accessed as an on-disk database (the default), or as an in-memory structure constructed (during configuration compilation or MTA initialization) from an on-disk flat text file. The on-disk database, if that is what is being used, is the `IMTA_GENERAL_DATABASE`, which must be generated with the `imsimta crdb` (UNIX or NT) utility from some site-supplied source text file. If an in-memory database structure is instead being used, then when the MTA configuration is compiled (or at MTA process initialization time, if a compiled configuration is not in use) the MTA reads the file referenced by the `IMTA_GENERAL_DATA` MTA Tailor file option, normally `IMTA_TABLE:general.txt`, and compiles it into an in-memory structure. Use of an in-memory "database" is normally recommended (for reasons of performance and reliability); however, do note that use of this in-memory "database" does require recompiling the configuration to get changes to the "database" (changes to the source text file) incorporated into the compiled configuration.

If a general database exists, it should be world readable to ensure that it operates properly.

Note that when wishing to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a general database substitution, the processing control metacharacter should be placed to the left of the general database substitution in the mapping table template; otherwise the "failure" of a general database substitution will mean that the processing control metacharacter will not be seen.

37.1.4.13 Mapping table substitutions, `$|...|`

A substitution of the form `$|mapping;argument|` is handled specially. The MTA looks for an auxiliary mapping table named *mapping* in the mapping file, and uses *argument* as the input (probe) to that named auxiliary mapping table. The named auxiliary mapping table must exist and must set the `$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 will be used as the output string.

Note that when wishing 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 will mean that the processing control metacharacter will not be seen.

37.1.4.14 Expression substitutions, `$`.'`

Expression substitution syntax is presently undocumented and subject to change.

37.1.4.15 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 an Oracle-supplied or customer-supplied routine. At run-time 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** arguments are 256 byte long character string buffers. On OpenVMS **argument** and **result** are passed by descriptor (a class S descriptor is used to ensure maximum compatibility); on Digital UNIX, Solaris, and NT, **argument** and **result** are passed as a pointer to a character string, (e.g., in C, as `char*`.) **arglength** and **reslength** are signed, 32 bit 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 will then replace 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 will be used unchanged as the output string.

Note that when wishing to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a site-supplied routine substitution, the processing control metacharacter should be placed to the left of the site-supplied routine substitution in the mapping table template; otherwise the "failure" of a mapping table substitution will mean that the processing control metacharacter will not be 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 `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.

See the [dns_verify callouts](#), the [check_memcache.so callout](#), and [check_metermaid callout](#) for examples of Oracle-provided such callout routines.

37.2 The mapping group

In Unified Configuration, the mapping group is not an option itself, but rather a grouping of mapping table entries defining a particular named [mapping table](#). For instance:

```
msconfig> set mapping:X-TEST.rule A $Y
msconfig# set mapping:X-TEST.rule B $Y
msconfig# set mapping:X-TEST.rule * $N
msconfig# show mapping:X-TEST
role.mapping:X-TEST.rule = A $Y
role.mapping:X-TEST.rule = B $Y
role.mapping:X-TEST.rule = * $N
msconfig# quit
```

This defines (but does not write -- does not save) a mapping table named X-TEST that returns a Y flag to an input probe of A or B, and returns a N flag to any other input probe. The legacy configuration (mappings file) equivalent would be:

```
X-TEST

A $Y
B $Y
* $N
```

37.3 Mapping operation

All MTA mapping tables are applied in a consistent way. What changes from one mapping to the next is the source of the input strings, and the use of the mapping output strings.

That is, a mapping operation always starts off with an input string and an MTA mapping table. The entries in the mapping table are scanned one at a time from first to last (top to bottom) in the order in which they appear in the table. The left hand side of each entry is used as pattern, and the input string is compared in a case-blind fashion with that pattern. 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, (that is, the input probe string "matched" the entry pattern), then the right hand 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.

37.4 Handling large numbers of mapping table entries

Sites that use very large numbers of entries in mapping tables should consider organizing their mapping tables to have a few generic wildcarded entries that [call out](#) to the [general database](#) for the specific lookups. It is much more efficient to have a few mapping table entries calling out to the general database for specific lookups than to have huge numbers of entries directly in the mapping table. (As a rule of thumb, certainly by the time a mapping table has reached about one hundred entries, it is worthwhile to consider whether the number of individual entries could be reduced by consolidating into more general entries that call out to the general database to check specific data.)

One case in particular is that some sites like to have per user controls on who can send and receive Internet e-mail. Such controls are conveniently implemented using a recipient 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 database](#) with mapping table entries structured to call out appropriately to the general database.

For instance, consider a mapping table:

```
ORIG_SEND_ACCESS

! Users allowed to send to Internet
!
*|adam@domain.com|*|tcp_local    $Y
*|betty@domain.com|*|tcp_local    $Y
! ...etc...
!
! Users not allowed to send to Internet
!
*|norman@domain.com|*|tcp_local    $NInternet$ access$ not$ permitted
*|opal@domain.com|*|tcp_local    $NInternet$ access$ not$ permitted
! ...etc...
!
! Users allowed to receive from the Internet
!
tcp_*|*|*|adam@domain.com        $Y
tcp_*|*|*|betty@domain.com        $Y
! ...etc...
!
! Users not allowed to receive from the Internet
!
tcp_*|*|*|norman@domain.com        $NInternet$ e-mail$ not$ accepted
tcp_*|*|*|opal@domain.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) would be as follows. Use general database entries of, say:

```
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
```

with an ORIG_SEND_ACCESS mapping table of:

```
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
```

Here 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 database probes with the `$C` and `$E` flags, as shown, is typical of mapping table callouts to the general database; see the [General database substitutions topic, under Mapping entry templates](#) for an additional discussion. For a discussion of the general database itself -- where it is located and how to build it--see [General database](#).

The above example showed a case of simple mapping table probes getting checked against general database entries. Mapping tables with much more complex probes can also benefit from use of the general database.

37.4.1 General database

The MTA's general database is available for site-specific uses. When a site could benefit from a database lookup from a [rewrite rule](#), from a [mapping table](#), or from a Sieve filter (see the [EXTLISTS](#) extension), the general database provides a simple, MTA-accessible place to store such site-specific data. (Note that the MTA also supports LDAP, memcache, and [MeterMaid](#) lookups, which could all be considered forms of "database" lookup. But what is meant here is more specifically an on-disk or in-memory, basic, key-value "database".)

In early versions of the MTA, the format of the general database was an on-disk database, built using the [imsimta crdb utility](#) based upon a flat text file input. As of JES MS 6.0, the option -- now preferred -- was introduced for storing the "database" directly in MTA process memory; use of such an "in memory database" is enabled via the [use_text_databases](#) MTA option. When enabled, such an "in memory database" is constructed from a flat text input file at the time of a [cnbuild](#) command being issued, or at process startup time (if no compiled configuration is used). The allowed format of the flat text input file is very similar whether an old `crdb` database is constructed, or whether a `use_text_databases` "in memory" database is constructed:

key value

one entry per line, with the key beginning in column one, one or more white space (SP or TAB) characters, and then the value on the right hand side.

When using a text, "in memory" general database (bit 0/value 1 of [use_text_databases](#) is set), each left hand side (key) may have a maximum of 128 characters, while each right hand side (value) may have a maximum of 1024 characters. If a key or value contains any space or tab character, such character must be backslash quoted, e.g.:

```
left\ hand\ side right-hand-side
```

Any TAB character found will be converted to a space for storage in the in-memory general database.

The `comment_chars` MTA option controls which characters (by default exclamation point and semicolon) in column one of a line are considered to indicate a comment line. The left angle character may be used to read another file into the general database text input file.

Note: The MTA options `general_data_size` and `string_pool_size_3` limit the overall size of the general database; these MTA options do not normally need to be manually adjusted.

For a `crdb` "on disk" database, the left hand side (the key) cannot contain spaces or tabs unless the `-quoted` switch is used; the maximum length of the key and value depend upon whether the `-long_records` switch is used.

37.5 When mapping table changes take effect

Changes to an MTA mapping table in general do not take effect until at a minimum the MTA configuration is reloaded; and if a compiled MTA configuration is in use, then the MTA configuration must be recompiled prior to the reload. Performing a restart of processes rather than a reload of the MTA configuration into already running processes is an alternative, though restarts of the Job Controller should be avoided (due to their disruptive effect on throughput) unless necessary.

37.6 Pre-defined mapping tables

The MTA has a number of pre-defined mapping tables. These include both mapping tables whose use (and names) are "hard-coded" in the MTA code for particular sorts of uses, as well as mapping tables which, while not "hard-coded", are included and referenced as part of a standard distribution. Such mapping tables include:

- `AUTH_ACCESS` mapping table
- `AUTH_REWRITE` mapping table
- `BURL_ACCESS` mapping table
- `CHARSET-CONVERSION` mapping table
- `COMMENT_STRINGS` mapping table
- `CONVERSIONS` mapping table
- `DISPOSITION_LANGUAGE` mapping table
- `ETRN_ACCESS` mapping table
- `FORWARD` mapping table
- `FROM_ACCESS` mapping table
- `GROUP_AUTH` mapping table
- `GROUP_TEMPLATES` mapping table
- `INTERNAL_IP` mapping table
- `IP_ACCESS` mapping table
- `LOG_ACTION` mapping table
- `MAC-TO-MIME-CONTENT-TYPES` mapping table
- `MAIL_ACCESS` mapping table
- `MESSAGE-SAVE-COPY` mapping table
- `MILTER_MACROS` mapping table
- `NOTIFICATION_LANGUAGE` mapping table
- `ORIG_MAIL_ACCESS` mapping table
- `ORIG_SEND_ACCESS` mapping table

- [PERSONAL_NAMES](#) mapping table
- [PORT_ACCESS](#) mapping table
- [REVERSE](#) mapping table
- [SEND_ACCESS](#) mapping table
- [SIEVE_EXTLISTS](#) mapping table
- [SPF_LOCAL](#) mapping table
- [TLS_ACCESS](#) mapping table

Note that to avoid conflicts with these, or to-be-defined-in-future, Oracle-provided mapping tables, it is recommended that all site-supplied mapping tables be given names beginning with X-, *e.g.*, X-whatever.

In addition to the pre-defined mapping tables listed above, the MTA has various alias options, alias file named parameters, and LDAP attributes (or rather in many cases, MTA options for selecting the name of an LDAP attribute) used to store mapping table names, and a facility for defining Sieve tests via mapping tables with a specific form of name; mapping tables named via such an alias file parameter or LDAP attribute or special Sieve test name syntax are used in the appropriate way by the MTA. Look for discussions of such mapping table use under topics including:

- [alias_auth_mapping](#)
- [alias_cant_mapping](#)
- [alias_deferred_mapping](#)
- [alias_direct_mapping](#)
- [alias_hold_mapping](#)
- [alias_nohold_mapping](#)
- [alias_moderator_mapping](#)
- [alias_sasl_auth_mapping](#)
- [alias_sasl_cant_mapping](#)
- [alias_sasl_moderator_mapping](#)
- The alias file named parameters [AUTH_MAPPING], [CANT_MAPPING], [HOLD_MAPPING], [NOHOLD_MAPPING], [MODERATOR_MAPPING], [SASL_AUTH_MAPPING], [SASL_CANT_MAPPING], and [SASL_MODERATOR_MAPPING]
- [ldap_url_result_mapping](#)
- [ldap_domain_attr_catchall_mapping](#)
- [FILTER_test](#) mapping tables
- The USERNAME_MAPPING SpamAssassin option discussed in [SpamAssassin_spamfilterN_config_file](#)

37.7 Testing mapping tables

Mapping tables are a very general MTA facility, routinely used for a wide variety of purposes, in a wide variety of contexts, and with a wide variety of potential inputs and outputs. Besides all of the MTA's own standard, [pre-defined mapping tables](#), sites may also incorporate their own, private mapping tables, for their own site-specific purposes. There is thus a "low-level" of basic mapping table output given a specific input which is a general aspect of mapping table behavior true for all mapping tables, and in contrast there is the "high-level" interpretation of mapping output which is performed by specific MTA components.

The MTA has two general, low-level command line test utilities, the `test -match` utility and the `test -mapping` utility, that can be used on any mapping table to test, respectively, the basic pattern matching and the low-level operation (the output-string and flags returned given some input-string) of mapping tables. But note that these utilities, and in particular

the `test -mapping` utility, do not provide any interpretation of the *meaning* or *effect* of a particular mapping table output-string, nor do they do any "validity check" on the input-string(s) supplied for testing. (And it is frequently the case that when a mapping table is not having the effect that you desire, that the real "problem" is that the input the MTA is feeding into the mapping table is not the input that you had anticipated.)

So when you wish to test for the *meaning* or *effect* of a mapping table, when you want to see what a mapping table does in real-world operation, these above-named utilities will not be useful. Instead, these utilities' intended purpose is for testing basic syntax of mapping tables, and they are best used primarily to answer syntax questions that arise when using fairly complex mapping tables. The `test -match utility` is useful when testing complex matching patterns, such as patterns that use "globs" or "IP subnet matching". The `test -mapping` utility is useful when testing mapping tables that contain complex recursive or iterative entries, or that include [callouts to routines](#).

In contrast, the *effects* of commonly-used addressing and access mapping tables such as the [FORWARD mapping table](#), the [REVERSE mapping table](#), the [FROM_ACCESS mapping table](#), or any of the [recipient access mapping tables](#) (`SEND_ACCESS`, `ORIG_SEND_ACCESS`, `MAIL_ACCESS`, `ORIG_MAIL_ACCESS`) can be conveniently tested using the `test -rewrite` utility. Note that when doing such testing, the utility's `-source_channel` and `-from` switches, and in the case of the `FROM_ACCESS`, `MAIL_ACCESS`, and `ORIG_MAIL_ACCESS` mapping tables also the `-applicationinfo` and `-transportinfo` switches, tend to be useful (indeed necessary) for proper testing.

(Aside: The `test -rewrite` utility can also be used to test posting access controls for mailing lists.)

37.7.1 Testing address access mapping tables

The `imsimta test -rewrite utility` can be useful in testing address access control mappings. Note that typically at least some of the utility's switches `-from`, `-source_channel`, `-destination_channel`, `-applicationinfo`, `-transportinfo`, and (new 6.2) `-sender` should be specified, in order to set relevant fields of the access mapping table probe and thus achieve meaningful testing. If an access control mapping table makes use of flag tests, then in order to properly test it, see also the `imsimta test -rewrite` utility's (new in 6.3) switches `-[no]esmtused`, `-[no]lmtused`, `-[no]proxyused`, `-[no]saslused`, `-[no]tlsused`. For instance, an `ORIG_SEND_ACCESS` mapping table of

```
ORIG_SEND_ACCESS
```

```
tcp_local|friendly@somewhere.com|1|AdamUser@acme.com      $Y
tcp_local|unwelcome@elsewhere.com|1|AdamUser@acme.com    $NGo$ away!
```

can be probed as follows:

```
# imsimta test -rewrite -from="friendly@somewhere.com" \
  -source=tcp_local -destination=1 AdamUser@acme.com
...
Submitted address list:
ims-ms
adam58@ims-ms-daemon (orig AdamUser@acme.com, inter AdamUser@acme.com, host
```

```
ims-ms-daemon) *NOTIFY-FAILURES* *NOTIFY-DELAYS*
```

Submitted notifications list:

```
# imsimta test -rewrite -from="unwelcome@elsewhere.com" \
  -source=tcp_local -destination=l AdamUser@acme.com...
```

Submitted address list:

```
Address list error -- Go away!: AdamUser@acme.com
```

Submitted notifications list:

If the `-debug` switch is also specified, then in addition to showing the effect of access control mapping table application, the output will also show the actual mapping table probe(s) constructed. For instance:

```
# imsimta test -rewrite -from="friendly@somewhere.com" \
  -source=tcp_local -destination=ims-ms AdamUser@acme.com -debug...
*** Debug output from submitting an envelope address:
...
12:27:18.86:      - orig_send_access mapping check: tcp_local|friendly@somewhere.com|l|AdamUser@acme.com
12:27:18.86:      - passed.
12:27:18.86:      - send_access mapping check: tcp_local|friendly@somewhere.com|ims-ms|adam58@ims-ms-daemon
12:27:18.86:      - passed.
...
Submitted address list:
  ims-ms
  adam58@ims-ms-daemon (orig AdamUser@acme.com, inter AdamUser@acme.com, host
  ims-ms-daemon) *NOTIFY-FAILURES* *NOTIFY-DELAYS*
```

Submitted notifications list:

37.8 Callout routines

The MTA's support for calling out to routines from [mapping tables](#) or [rewrite rules](#) is intended to allow sites to provide and use their own, site-written, routines. However, the MTA does ship with a few Oracle-provided routines as well.

37.8.1 check_memcache.so callout

New in 8.0, the `check_memcache.so` mapping callout can be used to access memcache from [mapping tables](#) or [rewrite rules](#). The callout is similar to the [check_metermaid.so callout](#), and provides a number of routines that can be called.

The general parameter format for all routines is:

```
[flags], [host[:port]][;host[:port]...], key[, value/lockout/quota][, timeout/test/quota_time]
0          1          2      3          4/5
```

The first four arguments common to all routines are:

- | | |
|------------------------------------|---|
| <p>flags (bit-encoded integer)</p> | <p>The flags parameter provides a number of flags that affect the entire option. The default is 0 if no value is specified. The bits and their meanings are shown in the table below.</p> |
| <p>host (string)</p> | <p>Host name of memcached server to use. The value specified by the memcache_host MTA option will be used if no host</p> |

is provided here. If a semicolon-separated list of host:port pairs is specified one will be chosen by applying a hash function to the key. This provides the means to distribute key-value pairs across multiple memcache servers. Note that the algorithm used is the same as for the [memcache Sieve extension](#).

port (integer 0-65535)

The memcached server port. The value specified by the [memcache_port](#) MTA option will be used if no port is provided here.

key (string)

The key associated with the entry being accessed.

Table 37.5 check_memcache.so flag parameter bit values

Bit(s)	Value(s)	Meaning
0-2	0-7	Debug level. Higher levels produce more debug output.
3	8	If set, causes all permanent failures to return as successful.
4	16	If set, causes an empty string to be returned regardless of what the callout did or didn't do.
5	32	Sets the penalize flag for throttle operations.
6	64	If set, hashes the provided key prior to use.
7	128	If set, normalizes the provided key to lower case
8	256	If set, don't add entry if it is missing. This flag applies to the ADJUST, ADJUST_AND_TEST, and THROTTLE routines.
9	512	If set, treat the entry (which must exist) as a throttle entry. This flag applies to the ADJUST and ADJUST_AND_TEST routines and must be used in order for these routines to handle throttle data.

Additional parameters are consumed by specific routines:

value (string or unsigned integer)

The value to be added, stored, replaced, etc.

timeout (unsigned integer)

Amount of time that the server should retain the entry, specified as an integer number of seconds. The default is 0, which means the entry should be retained indefinitely.

test

An unsigned integer preceded by a single character indicating the type of test to perform. Possible test types are: (1) <i - Callout succeeds if the result is less than i, (2) >i - Callout succeeds if the result is greater than i, and (3) =i - Callout succeeds if the result is equal to i.

quota_time, quota (unsigned integers)

Throttle parameters. See the MeterMaid documentation for details of the semantics of these parameters.

The available routines and their specific parameter formats are:

`ADD, flags, [host:[port]], key, value, timeout`

Adds an entry with the specified key, value and timeout. This routine will fail if an entry with the specified key is already present. An empty string is always returned.

`ADJUST, flags, [host:[port]], key, value[, timeout]`

Adjust the entry with the specified key by the amount specified by value. The entry must contain an unsigned decimal string and value must be an optionally signed integer. The specified entry will be created (with a value of 0) prior to adjustment if it doesn't already exist. The adjusted value is returned as an unsigned decimal string. The timeout value is only used if the entry has to be created.

`ADJUST_AND_TEST, flags, [host:[port]], key, value, test[, timeout]`

Adjust the entry with the specified key as ADJUST would, then test the result with the specified test. An empty string is always returned.

`APPEND, flags, [host:[port]], key, value`

Append the specified value to the entry with the specified key. This routine will fail if an entry with the specified key is not already present. An empty string is always returned.

`FETCH, flags, [host:[port]], key`

Return the value of the entry with the specified key. The callout fails if no entry with the specified key is present.

`PREPEND, flags, [host:[port]], key, value`

Prepend the specified value to the entry with the specified key. This routine will fail if an entry with the specified key is not already present. An empty string is always returned.

`REMOVE, flags, [host:[port]], key[, lockout]`

Remove the entry with the specified key. Lockout, if present, is an unsigned integer specifying the amount of time to "lock" the key - during that time attempts to add an entry with that key will fail. A lockout default of 0 is the default and causes no lockout to occur. This routine will fail if an entry with the specified key is not already present. An empty string is always returned.

`REPLACE, flags, [host:[port]], key, value, timeout`

Update value and timeout of the entry with the specified key. This routine will fail if an entry with the specified key is not already present. An empty string is always returned.

`STORE, flags, , [host:[port]], key, value, timeout`

Creates a new entry or updates an existing entry with the specified key, value, and timeout.

```
TEST, flags, , [host:[port]], key, test
```

Tests the value of the entry with the specified key.

```
THROTTLE, flags, , [host:[port]], key, quota, quota_time
```

Implements the MeterMaid throttle capability. See the MeterMaid documentation for details of throttle semantics. Note that since there is no server-side awareness of entry semantics the quota and quota_time parameters must be specified in every throttle call in case the entry needs to be created. If the entry already exists the parameter values will be checked against the corresponding values stored in the entry. The call will fail if this check fails.

This callout has also been enhanced to work with the mapping's \$. facility in order to be able to handle temporary errors in a sensible fashion.

37.8.2 check_metermaid callouts

The check_metermaid.so mapping callout can be used to access MeterMaid from [mapping tables](#) or [rewrite rules](#). A number of entry points are provided.

```
adjust, table, key, adjustment
```

(New in MS 7.0u2) Similar to store, but adjustment is treated as a delta value. It can be specified as integer, +integer, or -integer. If the key doesn't currently exist, it is presumed to be 0 and the value is set to whatever the adjustment is. Succeeds by default, fails if an error occurs.

```
adjust_and_test, table, key, adjustment, test-expr
```

(New in MS 7.0u2) This combines the adjustment with a test. The return value is the same as test.

```
fetch, table, key
```

(New in MS 7.0u2) Returns a string from that key's value. Fails if an error occurs or the key is not currently set.

```
greylisting, table, key
```

```
query, table, key
```

```
remove, table, key
```

(New in MS 7.0u2) Removes the key/value pair specified by the key from the table. Return success by default, and fail if an error occurs.

store, table, key, value

(New in MS 7.0u2) Sets the value for the specified key in that table. Returns success if completed successfully, and fails if not.

test, table, key, test-expr

(New in MS 7.0u2) *test-expr* is a simple expression that gives the test to be done. It consists of an operator and a value. The operator can be one of =, >, >=, <, or <=. The value should be an integer.

This returns success if the *test-expr* returns true, and fails if it's false or an error occurs. (This flexibility permits control of whether the default should be to pass, or to fail.)

throttle, table, key

Apply throttling.

New in MS 8.0, `check_metermaid.so` supports use of multiple MeterMaid servers, and also supports [SSL](#) for communication.

37.8.3 dns_verify callouts

The `dns_verify.so` library provides a collection of mapping callouts that can be used to validate and/or resolve domains names or IP addresses via the DNS or local host tables. (Exactly what sources are used, and in what order, is controlled at the operating system level, usually by settings in `/etc/resolv.conf`.) Three basic types of operations are provided:

- DNS A/AAAA record and host table lookups for domain names.
- DNS PTR record and host table lookups for IP addresses.
- TXT record lookups for checking IP addresses against blacklists.

For example, `dns_verify.so` can be used to verify that an entry in DNS or host tables exists for the domain used in the SMTP MAIL FROM: command, or to look up an IP address in a blacklists supplied by such services as MAPS and ORBS. The message can be rejected or accepted based on the presence or absence of a corresponding DNS record.

IMPORTANT NOTE: Performing DNS existence checks may result in the rejection of some valid messages. For instance, this could include mail from legitimate sites that simply have not yet registered their domain name, or during periods of bad information in DNS.

The `dns_verify.so` library has 5 routines that can be called:

- `dns_verify`
- `dns_verify_ptr` (new in 7.0.5.34)
- `dns_verify_domain`
- `dns_verify_domain_port`
- `dns_verify_domain_warn`

These routines are each described in the sections below.

Note that your mapping tables with `dns_verify.so` callouts can be tested by using the `imsimta test -mapping` utility.

37.8.3.1 The `dns_verify`

The `dns_verify` routine does a name lookup in the local host tables and an A/AAAA lookup in the DNS. One possible use for this is to check to make sure the domain from the SMTP `MAIL FROM:` command actually exists. Any mapping table action can be taken if the lookup succeeds, fails, or returns an error.

The `dns_verify` routine's argument is four strings separated by "|", as follows:

```
domainname[|success[|failure[|unknown]]]
```

<code>domainname</code>	The name to look up in the DNS and local host tables.
<code>success</code> (optional)	If specified, it is the mapping table string to return if <code>domainname</code> is found. If not specified, the default is "\$Y".
<code>failure</code> (optional)	If specified, it is the mapping table string to return if <code>domainname</code> is not found. If not specified, the default is "\$N".
<code>unknown</code> (optional)	If specified, it is the mapping table string to return if there was a temporary DNS failure during the lookup operation. If not specified, and the <code>success</code> string was specified, that string is used. If neither are specified, the default is "\$Y".

Note that in the mapping table any \$'s you wish to return need to be doubled. For example, to specify "\$Y", you need to put in "\$\$Y".

An alternate separator can be used instead of "|". To specify an alternate separator, insert it as the first character of the routine's argument. For example, to specify "+" as the separator, use the following syntax:

```
+domainname+success+failure+unknown
```

The `success`, `failure`, and `unknown` strings can contain the following format characters:

String	Value
<code>%a</code>	If successful, the <code>%a</code> substitutes the IP address returned by the lookup operation. An empty string is returned if the domain name exists but doesn't have an associated IP address.
<code>%e</code>	If the lookup is not successful, <code>%e></code> substitutes the error message associated with the lookup.
<code>%n</code>	If successful, <code>%n</code> substitutes the primary name for <code>domainname</code> . An empty string is returned if the domain name exists but doesn't have an associated IP address.

The following example shows a simple `SEND_ACCESS` mapping table entry to verify that the sender's hostname exists in the DNS or local host tables:

```
SEND_ACCESS
```

```
*tcp_*|*@*|*|* \
$C$[IMTA_LIB:dns_verify,dns_verify,$3|$$Y|$$NInvalid$ host:$ $$3$-$ %e]$E
```

The following example shows a PORT_ACCESS mapping table entry that performs a check against a hypothetical DNS blacklist dnsbl.example.net

```
PORT_ACCESS
```

```
TCP|*|25|*.*.*.*|* \
$C$[IMTA_LIB:dns_verify,dns_verify,\
$4.$3.$2.$1.dnsbl.example.net|$$N500$ IP$ blacklisted|$$Y
```

37.8.3.2 The dns_verify_ptr

The `dns_verify_ptr` routine does an IPv4/IPv6 address lookup in the DNS and/or host tables. Any mapping table action can be taken if the lookup succeeds, fails, or returns an error.

The `dns_verify` routine's argument is four strings separated by "|", as follows:

```
ip-address[ |success[ |failure[ |unknown]]]
```

<code>ip-address</code>	The IPv4/IPv6 address to be looked up, without any enclosing brackets or prefixes.
<code>success (optional)</code>	If specified, it is the mapping table string to return if <code>ip-address</code> is found. If not specified, the default is "\$Y".
<code>failure (optional)</code>	If specified, it is the mapping table string to return if <code>ip-address</code> is not found. If not specified, the default is "\$N".
<code>unknown (optional)</code>	If specified, it is the mapping table string to return if there was a temporary DNS failure during the lookup operation. If not specified, and the <code>success</code> string was specified, that string is used. If neither are specified, the default is "\$Y".

The `dns_verify_ptr` routine supports the same alternate delimiter and substitution strings as the `dns_verify` routine described above.

37.8.3.3 dns_verify_domain and dns_verify_domain_port

The `dns_verify_domain` and `dns_verify_domain_port` routines are used to perform queries for DNS entries with well-defined blacklist semantics and return pre-defined success, failure, and unknown messages. The same operation can be performed using the `dns_verify` routine, but with more complicated setup.

The `dns_verify_domain_port` routine is designed for use in the PORT_ACCESS mapping table. The `dns_verify_domain` routine is used in the MAIL_ACCESS, SEND_ACCESS, and similar mapping tables.

The `dns_verify_domain` and `dns_verify_domain_port` routines perform the same type of action as the [dns_verify_domain](#) dispatcher option. Using the routine allows you more control over which connections trigger the DNS lookups.

The `dns_verify_domain` and `dns_verify_domain_port` routines' argument is three strings separated by ",", as follows:

```
ip-address, domainname[, text-string]
```

<code>ip-address</code>	The IP address that you want to check against the blackhole list
<code>domainname</code>	The name of the blackhole list to check against, e.g., <code>blackholes.mail-abuse.org</code>
<code>text-string (optional)</code>	If specified, it is the text to return if no TXT record is available. If not specified, the default is "No Error Text Available".

The `dns_verify_domain` and `dns_verify_domain_port` routines check the given list for the IP address. For example, if `ip-address` is `127.0.0.2`, and `domainname` is `bl.spamcop.net`, `dns_verify_domain` and `dns_verify_domain_port` looks up the following name: `2.0.0.127.bl.spamcop.net`. They first first look up the TXT record for that name, and if it is not available, they look up the A record.

The following examples illustrate the use of these routines.

```
MAIL_ACCESS
```

```
TCP|*|25|*|*|*|*|tcp_local|*|*|* \
$C$[IMTA_LIB:dns_verify,dns_verify_domain,$1,bl.spamcop.net]$E
```

```
PORT_ACCESS
```

```
TCP|*|25|*|*|* \
$C$[IMTA_LIB:dns_verify,dns_verify_domain_port,$1,bl.spamcop.net]$E
```

The approximate equivalent of the previous `MAIL_ACCESS` example using the `dns_verify` routine would be something like:

```
MAIL_ACCESS
```

```
TCP|*|25|*|*|*|*|*|tcp_local|*|*|* \
$C$[IMTA_LIB:dns_verify,dns_verify,+$4.$3.$2.$1.bl.spamcop.net+\
$N$X5.7.1|Blocked$ -$ see$ http://spamcop.net/bl.shtml?$$1.$$2.$$3.$$4+$$Y]$E
```

37.8.3.4 dns_verify_domain_warn

The `dns_verify_domain_warn` routine performs the same DNS lookup as the `dns_verify_domain` and `dns_verify_domain_port` routines, but instead of rejecting the message if the DNS entry exists, it adds a new header line to the message. The `dns_verify_domain_warn` routine can be used in any of the sender or recipient access mapping tables.

The `dns_verify_domain_warn` routine's argument is four strings separated by ",", as follows:

Chapter 38 Message conversions

38.1 Conversion channel	38-1
38.1.1 CONVERSIONS mapping table: Selecting message traffic for conversion processing	38-2
38.1.2 Conversion channel definition	38-5
38.1.3 Conversion control	38-6
38.2 Conversion tags	38-15
38.3 Character set conversion and message reformatting	38-16
38.3.1 CHARSET-CONVERSION mapping table	38-16
38.3.2 Message reformatting	38-20
38.3.3 Relabelling MIME header lines	38-25
38.3.4 Service conversions	38-26
38.4 Interactions between conversions and character set conversions	38-28

There are two broad categories of message conversions in the MTA, controlled by two corresponding [mapping tables](#) and the MTA [conversions](#). In Unified Configuration, the mappings are stored under the [mapping](#) MTA option, and the MTA conversion entries are stored under the [conversions](#) MTA option; in legacy configuration, the MTA mapping tables are stored in the `mappings` file, and the MTA conversion entries are stored in the `conversions` file.

The first category of message conversion is that of character set, formatting, and labelling conversions performed internally by the MTA. The application of such conversions is controlled by the [CHARSET-CONVERSION mapping table](#). CHARSET-CONVERSIONS are discussed in [Character set conversion and message reformatting](#).

The second category of message conversion is that of conversion of message attachments using external, third-party programs and site-supplied procedures, such as document convertors. The application of such conversions is controlled by the [CONVERSIONS mapping table](#), and messages requiring such conversions are thereby routed through the MTA's [conversion channel](#); the conversion channel executes the site-specified external conversion procedures.

Note that the MTA [conversions](#) options (stored in the `conversions` file in legacy configuration), are used to specify the details of [CONVERSIONS mapping table](#) triggered external conversions and to specify the details of some internal [CHARSET-CONVERSION mapping table](#) triggered conversions.

38.1 Conversion channel

The conversion channel performs arbitrary body-part-by-body-part conversions on messages flowing through the MTA. Any subset of MTA message traffic can be selected for conversion and any set of programs or command procedures can be used to perform conversion processing. (The MTA's native conversion facilities are fairly limited, so the ability to call external convertors is crucial.) A special "database" (stored as the [conversions MTA option](#) in Unified Configuration, or in the `conversions` file in legacy configuration) is consulted to choose an appropriate conversion for each body part.

For instance, third party document convertors or virus scanning software may be hooked in for automatic execution via the conversion channel. Or sites may develop their own custom applications to hook in via the conversion channel.

Because the conversion channel is intended for "intermediate" processing of messages, the MTA has a special sort of routing available for it, whereby messages are routed without affecting the recipient address(es); see the [CONVERSIONS mapping table](#). This special sort of routing used to route messages through the conversion channel without modification to the actual addresses can also be used for other purposes: to route through third party channel programs, or to route out to third party spam/virus SMTP hosts (that will then relay messages back to the MTA). See the discussion of [alternate channel routing via the CONVERSIONS mapping](#)

38.1.1 CONVERSIONS mapping table

Although conversion processing is done using a regular MTA channel program, under normal circumstances this channel is never specified directly either in an address or in an MTA rewrite rule. Instead, the MTA controls routing to the conversion channel via the CONVERSIONS mapping table.

In legacy configuration, the CONVERSIONS mapping table (like all mapping tables), was stored in the `mappings` file. In Unified Configuration, the CONVERSIONS mapping table is stored as the settings under either a `role.mapping:CONVERSIONS` option or a `instance.mapping:CONVERSIONS` option. In Unified Configuration, most often creation or modification of such a CONVERSIONS mapping table is performed using the `edit` command of the `msconfig`, to edit any or all mapping tables in a format like the legacy `mappings` file; *e.g.*:

```
msconfig> edit mappings
```

Or from within `msconfig` the CONVERSIONS mapping table can be created line-by-line:

```
msconfig> set mapping:CONVERSIONS.rule "IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT" Yes
msconfig# set mapping:CONVERSIONS.rule "IN-CHAN=ims-ms;OUT-CHAN=tcp_local;CONVERT" Yes
msconfig# set mapping:CONVERSIONS.rule * No
```

Note that CONVERSIONS mapping rules often include a special character such as the equal sign, `=`, in either or both of the template (left hand side of the rule) or pattern (right hand side of the rule), thus often require quoting of template and/or pattern when being set at the `msconfig` command line.

As the MTA processes each message it probes the CONVERSIONS mapping (if one is present) with a string of the default form

```
IN-CHAN=source-channel;OUT-CHAN=destination-channel;CONVERT
```

where `source-channel` is the source channel from which the message is coming and `destination-channel` is the destination channel to which the message is heading. New in JES MS 6.3, setting bit 1 (value 2) of the [include_conversiontag MTA option](#) will cause the probe to instead have the form

```
IN-CHAN=source-channel;OUT-CHAN=destination-channel;TAG=tag-list;CONVERT
```

where `tag-list` is a comma-separated list of any [conversion tags](#) present on the message. Note that multiple conversion tags may be present; multiple tags will be included as a comma-separated list. The total, combined length of such tags (that is, the length of the list, including

the commas) is limited to 256 characters. (As with all mapping tables, the overall probe length is limited to 1024 characters.)

New in the 8.0 release, setting bit 6 (value 64) of the `include_mtpriority` MTA option will case an additional, compound field to be appended to the CONVERSIONS mapping table probe, immediately after any "TAG=" clause.

If the probe matches the pattern (left hand side) of a CONVERSIONS mapping table entry, then the resulting string (right hand side of the mapping entry) should be a comma-separated list of keywords. Usually either just the keyword "Yes" or "No" is specified. If "Yes" is produced, the MTA will divert the message from its regular destination to the conversion channel. (By also specifying `Channel=channel-name`, the message can be diverted to some [alternate channel](#) rather than to the regular conversion channel.) If the message has a [conversion tag](#) set, note that the "T" flag will be set, and this can be tested for using a `$:T` test in the output string. If either "No" is produced or no match is found, the message will be queued to the regular destination channel.

Some less commonly used, additional template keywords, similar to those available for the [CHARSET-CONVERSION mapping table](#), are also available as shown below.

Table 38.1 Additional CONVERSION mapping keywords

Keyword	Action
Always	Force conversion even when the "conversion" channel is the same as the source channel; while not desirable when the actual conversion channel (or any other "intermediate channel") is being used, "Always" can be useful when an "alternate conversion channel", specified via a "Channel= <i>channel-name</i> " clause, is used to select some other sort of channel, such as a <code>tcp_*</code> channel
Appledouble	Convert other MacMIME formats to Appledouble format
Applesingle	Convert other MacMIME formats to Applesingle format
BASE64	Switch MIME encodings to BASE64
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
Channel= <i>channel-name</i>	Route through the alternate channel <i>channel-name</i> rather than the regular conversion channel
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
Pathworks	Convert message to Pathworks Mail format
Preprocess	(New in JES MS 6.3) Perform any configured charset conversion <i>before</i> routing to the conversion channel
QUOTED-PRINTABLE	Switch MIME encodings to QUOTED-PRINTABLE

CONVERSIONS mapping table:
 Selecting message traffic for
 conversion processing

Record,Text	Line wrap text/plain parts at 80 characters
Record,Text= <i>n</i>	Line wrap text/plain parts at <i>n</i> characters
RFC1154	Convert message to RFC 1154 format
Thurman	Convert some non-standard "attachments" to MIME 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

For example, suppose messages coming in from the Internet and destined to the Message Store via either an [ims-ms](#) or [tcp_lmtpcs*](#) channel require conversion processing. The following mapping would then be appropriate:

CONVERSIONS

```
IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT      Yes
IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;CONVERT  Yes
IN-CHAN=*;OUT-CHAN=*;CONVERT                   No
```

In Unified Configuration, `msconfig's` `edit` command could show the `CONVERSIONS` mapping as above. Alternatively,

```
msconfig> show mapping:CONVERSIONS.*
role.mapping:CONVERSIONS.rule = IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT Yes
role.mapping:CONVERSIONS.rule = IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;CONVERT Yes
role.mapping:CONVERSIONS.rule = IN-CHAN=*;OUT-CHAN=*;CONVERT No
```

38.1.1.1 Alternate channel routing via the CONVERSIONS mapping

One of the `CONVERSIONS` mapping table template keyword clauses is `Channel=channel-name`, causing routing (if the `CONVERSIONS` mapping table pattern matched the probe) through the specified `channel-name` without any change to the message's recipient addresses; see the [CONVERSIONS mapping table](#). This `CONVERSIONS` mapping table facility to cause routing via some alternate channel (without alteration of a message's recipient addresses) turns out to be extremely useful for a number of scenarios. Originally conceived as a convenience for hooking in third-party or site-developed message processing channels, *e.g.*,

CONVERSIONS

```
IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT \
Yes,Channel=third-party-processing-channel
```

alternate conversion channel configuration also functions perfectly well to route messages out via SMTP to some [third-party spam/virus filtering SMTP box](#) (presumably itself configured to send processed messages back to the MTA to further delivery). For instance, if the MTA has been configured with a special `tcp_*` channel for sending to, and receiving from, a third-party spam/virus filtering SMTP host, with rewrite rule:

```
! Rewrite rule to recognize source IP of virus scanner box and "switch"
! messages coming from it to be considered to come in tcp_virusscanner channel
```

```
!
[IP-address-of-virusscanner]    $E$R$U%$H@TCP-VIRUSSCANNER-DAEMON
```

and channel definition:

```
! Channel for sending to virus scanner box.
! Set with "allowswitchchannel" so that it can also be considered the source
! channel for messages coming back in from virus scanner box.
!
tcp_virusscanner smtp daemon host-name-of-virusscanner \
  allowswitchchannel ...additional-keywords...
TCP-VIRUSSCANNER-DAEMON
```

then a corresponding CONVERSIONS mapping table on a front end MTA might be along the lines of:

```
CONVERSIONS
```

```
IN-CHAN=tcp_local;OUT-CHAN=tcp_intranet;CONVERT Yes,Channel=tcp_virusscanner
```

to cause all messages coming directly in from the Internet, destined for an internal host, to get a "side hop" through the virus scanner box. (Note that for users hosted-on-this-MTA, recipients on `ims-ms` or `tcp_lmtpcs` channel, the "side hop" routing should be configured via the [aliasdetourhost](#) channel option as the timing of its effect has better implications for local-to-this-host users; but "alternate conversion channel" routing is generally used in conjunction with `aliasdetourhost` in order to handle the cases of messages to more remote users.)

Such "alternate conversion channel" routing has also found an important application in causing special routing of notification messages; see [notification message routing](#).

38.1.2 Conversion channel definition

A conversion channel must be present in the MTA's configuration in order for conversions to be performed. In Unified Configuration, a conversion channel is configured automatically, while in legacy configuration the post-installation configuration step normally caused an appropriate conversion channel definition to be included in the MTA configuration file. The channel definition should have the general form:

```
conversion
CONVERSION-DAEMON
```

or as displayed in Unified Configuration:

```
msconfig> show channel:conversion.*
role.channel.conversion.official_host_name = conversion-daemon
```

As of MS 8.0, use of the [receivedstate](#) channel option is recommended:

```
conversion receivedstate "content"
CONVERSION-DAEMON
```

or in Unified Configuration:

```
msconfig> set channel:conversion.receivedstate content
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the conversion channel. Something like

```
conversion                $U%conversion.localhostname@conversion-daemon
conversion.localhostname  $U%conversion.localhostname@conversion-daemon
```

where `localhostname` is the name of this MTA system, will provide the necessary functionality. Note that in Unified Configuration, such rewrite rules would typically be automatically generated, and make use of the `$D` and `&/IMTA_HOST/` and `&/IMTA_DEFAULTDOMAIN/` [substitutions](#), to appear as:

```
msconfig> show rewrite.rule * *conversion*
role.rewrite.rule = conversion $U%conversion.domain.com@conversion-daemon
role.rewrite.rule = conversion.&/IMTA_HOST/ $U%conversion.domain.com@conversion-daemon
```

Once such rewrite rules are present, then addresses of the form

```
user%host@conversion.localhostname
```

will be routed through the conversion channel regardless of what the [CONVERSIONS mapping table](#) says.

38.1.3 Conversion control

The actual conversions performed by the conversion channel are controlled by rules specified in the MTA conversions file (legacy configuration) or [conversions](#) MTA option (Unified Configuration). As of MS 7.0, the conversions file is (symbolically) `IMTA_TABLE:conversions`, *i.e.*, `SERVERROOT/config/conversions`. (In prior versions, the conversions file was located via the IMTA tailor file option `imta_conversion_file`, normally pointing to `/opt/SUNWmsgsr/config/conversions`.)

Note: Even in Unified Configuration, it is most convenient to think of the conversions option as a file, accessed and modified via the command

```
msconfig> EDIT CONVERSIONS
```

so from here on, references will be simply to "the conversions file", even for a Unified Configuration.

The MTA conversions file is a text file containing entries in a format that is modelled after MIME Content-Type: parameters. Each entry consists of one or more lines grouped together; each line contains one or more "name=value;" parameter clauses. Quoting rules conform to MIME conventions for Content-Type: header line parameters. Every line except the last must end with a semicolon. Entries are terminated by either a line that does not end in a semicolon, one or more blank lines, or both. For example, the following entry specifies that application/x-ddif parts in messages sent out to the Internet should be converted to PostScript:

```
out-chan=l; in-type=application; in-subtype=x-ddif;
  out-type=application; out-subtype=postscript; parameter-copy-0=*;
```

```
command="ddifps $INPUT_FILE $OUTPUT_FILE"
```

38.1.3.1 Conversion entry scanning and application

The conversion channel processes each message routed through it, part by part. The header of each part is read and its Content-Type: and other header information is extracted. (Note that composite media types, that is, MULTIPART/* or MESSAGE/* "parts", are not made available per se to the conversion channel, though any component discrete body parts within such composite types are made available to the conversion channel for potential processing.)

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 remainder of the entry is performed. Note that an entry must include an IN-TYPE clause in order to match. More specifically, the matching checks:

- if the IN-CHAN and OUT-CHAN parameters match the channels¹ through which the message is passing;
- and if the PART-NUMBER matches the structured part number² of the message part;
- and if all of the IN-PARAMETER-NAME, IN-PARAMETER-VALUE, IN-SUBTYPE, and IN-TYPE, parameters match the Content-Type: of the message part;
- and if all of the IN-DISPOSITION, IN-DPARAMETER-NAME, and IN-DPARAMETER-VALUE parameters match the Content-Disposition of the message part;
- and if the IN-DESCRIPTION matches the Content-Description of the message part;
- and if the IN-SUBJECT, IN-MESSAGE-CONTEXT, IN-A1-TYPE, and IN-A1-FORMAT of the headers of the immediately enclosing message (message/rfc822 part) match those immediately enclosing the message part;
- and (as of MS 7.0.5) if the IN-ENCODING matches the Content-transfer-encoding of the message part.

Only if all specified parameters match is the entry considered to match. Scanning terminates once a matching entry has been found or all entries have been exhausted. If no entry matches, no conversion is performed.

If the matching entry specifies DELETE=1, then the message part is deleted. Otherwise, the command specified by the COMMAND parameter is executed.

Once an entry with a COMMAND parameter has been selected, the body part is extracted to a file. The converter execution environment is prepared as specified by the PARAMETER-SYMBOL-n parameters and DPARAMETER-SYMBOL-n parameters. Finally, a subprocess is created to run the command specified by the COMMAND parameter. The command should perform the necessary conversion operation, reading the file specified by the environment variable (UNIX) and producing the file specified by the environment variable (UNIX).

On UNIX, the command may optionally set options in the OUTPUT_OPTIONS file to pass information back to the conversion channel.

Conversion operations are terminated and no conversion is performed if the forked command returns an error.

If the command succeeds, the resulting output file is read as specified by the OUT-MODE parameter and a new body part containing the converted material is constructed according

to the OUT-ENCODING, OUT-PARAMETER-NAME-n, OUT-PARAMETER-VALUE-n, OUT-SUBTYPE, OUT-TYPE, OUT-DESCRIPTION, OUT-DISPOSITION, OUT-DPARAMETER-VALUE-n parameters.

This process is repeated for each part of the message until all parts have been processed.

See [Conversion entry parameters](#) for a complete list of the parameters available to conversion entries.

¹ The source channel and destination channel are normally the original source channel and original destination channel prior to the [CONVERSIONS mapping table](#) applying and forcing a "hop" through the conversion channel: that is, the conversion channel itself is not normally the IN-CHAN or OUT-CHAN. However, see the [original_channel_probe](#) MTA option, and [explicit routing of an address through the conversion channel](#) for exceptions.

² The structured part number is the message part number as it would appear in PMDF MAIL. That is, a multipart message has outer "level" parts counted starting from 1, and with a part, if it is a multipart itself, the subparts count starting from 1, and so on for additional "levels". So a multipart with no additional sublevels may have part numbers 1, 2, 3, etc., while a multipart whose second part is itself a multipart, might have part numbers 1, 2.1, 2.2, etc., 3, etc..

38.1.3.2 Conversion entry parameters

The rule parameters currently provided are shown listed by functional groups in [Available conversion parameters](#).

Parameters not listed in this table (below) are ignored.

Table 38.2 Available conversion parameters, grouped by functionality

Parameter	Meaning
Command parameters	
CALL	Routine to call. The argument takes the form <i>image routine argument</i> .
COMMAND	Command to execute to perform conversion. This parameter (or a CALL or DELETE parameter) is normally required; if no command is specified, the entry is ignored during conversion channel processing . ¹
DELETE	0 or 1. If this flag is set, the message part will be deleted. (If this is the only part in a message, then a single empty text part will be substituted.)
RELABEL	0 or 1. This flag causes an entry to be ignored during conversion channel processing . However, if this flag is 1, then MIME header relabelling is performed during character set conversion . ³
SERVICE-CALL	Routine to call to perform service conversion . The argument takes the form <i>image routine argument</i> . Note that this flag causes an entry to be ignored during conversion channel processing ; SERVICE-CALL entries are instead performed during character set conversion processing . ²
SERVICE-COMMAND	The command to execute to perform service conversion . This parameter (or SERVICE-CALL) is required in order to perform a service conversion; if no command (or call) is specified, the entry is ignored for that phase of processing. 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 . ²
Matching parameters	

ATTACHMENT-NUMBER	Sequential number of the part, starting at number 0 for the first part of the message. Compare with PART-NUMBER, which is the MIME structured number for the part.
IN-A1-FORMAT	Input A1-Format: value from the enclosing message/rfc822 part.
IN-A1-TYPE	Input A1-Type: value from the enclosing message/rfc822 part.
IN-CHAN	Input channel to match for conversion (wildcards allowed). The conversion specified by this entry will only be performed if the message is coming from the specified channel.
IN-CHANNEL	Synonym for IN-CHAN.
IN-DESCRIPTION	Input MIME Content-Description: value.
IN-DISPOSITION	Input MIME Content-Disposition.
IN-DPARAMETER-DEFAULT-n	Input MIME Content-Disposition parameter value default if parameter is not present. This value is used as a default for the IN-DPARAMETER-VALUE-n test when no such parameter is specified in the body part.
IN-DPARAMETER-NAME-n	Input MIME Content-Disposition parameter name whose value is to be checked; $n = 0, 1, 2, \dots$
IN-DPARAMETER-VALUE-n	Input MIME Content-Disposition parameter value that must match corresponding IN-DPARAMETER-NAME (wildcards allowed). The conversion specified by this entry is only performed if this field matches the corresponding parameter in the body part's Content-Disposition: parameter list.
IN-ENCODING	(New in MS 7.0.5) Input Content-transfer-encoding: value.
IN-LANGUAGE	Input Content-language: value.
IN-MESSAGE-CONTEXT	Input Message-context: value.
IN-PARAMETER-DEFAULT-n	Input MIME Content-Type parameter value default if parameter is not present. This value is used as a default for the IN-PARAMETER-VALUE-n test when no such parameter is specified in the body part.
IN-PARAMETER-NAME-n	Input MIME Content-Type parameter name whose value is to be checked; $n = 0, 1, 2, \dots$
IN-PARAMETER-VALUE-n	Input MIME Content-Type parameter value that must match corresponding IN-PARAMETER-NAME (wildcards allowed). The conversion specified by this entry is only performed if this field matches the corresponding parameter in the body part's Content-Type: parameter list.
IN-SUBJECT	Input Subject from enclosing message/rfc822 part.
IN-SUBTYPE	Input MIME subtype to match for conversion (wildcards allowed). The conversion specified by this entry is only performed if this field matches the MIME subtype of the body part.
IN-TYPE	Input MIME type to match for conversion (wildcards allowed). The conversion specified by this entry is only performed if this field matches the MIME type of the body part.

OUT-CHAN	Output channel to match for conversion (wildcards allowed). The conversion specified by this entry will only be performed if the message is destined for the specified channel.
OUT-CHANNEL	Synonym for OUT-CHAN.
PART-NUMBER	Dotted integers, <i>e.g.</i> , <i>a.b.c...</i> The part number of the MIME body part.
TAG	Input conversion tag must match; such conversion tags might have been set in various ways, including via a mailing list [CONVERSION_TAG] named parameter , or via an alias_conversion_tag alias option, or in direct LDAP mode set via a user's own mailConversionTag attribute or via the user's domain's mailDomainConversionTag attribute, or via Sieve filter conversion tag actions .
Conversion script environment parameters	
DPARAMETER-SYMBOL-n	Content-disposition parameters to convert to environment variables if present; $n = 0, 1, 2, \dots$ Takes as argument the name of the MIME parameter to convert, as matched by an IN-DPARAMETER-NAME-m clause. Each DPARAMETER-SYMBOL-n is extracted from the Content-Disposition: parameter list and placed in an environment variable of the same name prior to executing the converter.
MESSAGE-HEADER-FILE	0, 1, or 2. If set to 1, the original headers of the immediately enclosing message part are written to the file represented by the MESSAGE_HEADERS symbol. If set to 2, the original headers of the message as a whole (the outermost message headers) are written to MESSAGE_HEADERS. As of MS 6.1, in the MESSAGE-HEADER-FILE=2 case, besides the original headers, the envelope information will also be written, in the form of an X-Envelope-from: header line and an X-Envelope-to: header line.
ORIGINAL-HEADER-FILE	0 or 1. If set to 1, the original headers of the enclosing part are written to the file represented by the INPUT_HEADERS symbol.
PARAMETER-SYMBOL-n	Content-Type parameters to convert to environment variables if present; $n = 0, 1, 2, \dots$ Takes as argument the name of the MIME parameter to convert, as matched by an IN-PARAMETER-NAME-m clause. Each PARAMETER-SYMBOL-n is extracted from the Content-Type: parameter list and placed in an environment variable of the same name prior to executing the converter.
Override/output parameters	
DPARAMETER-COPY-n	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; $n = 0, 1, 2, \dots$ Takes as argument the name of the MIME parameter to copy, as matched by an IN-DPARAMETER-NAME-m clause. Wildcards may be used in the argument. In particular, an argument of * means to copy all the original Content-Disposition: parameters.
OUT-A1-FORMAT	Output A1-Format.
OUT-A1-TYPE	Output A1-Type.
OUT-DESCRIPTION	Output MIME Content-Description if it is different than the input MIME Content-Description.
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> .
OUT-MODE	Mode in which to read the converted file. This should be one of: BLOCK, RECORD, RECORD-ATTRIBUTE, TEXT.
OUT-ENCODING	Encoding to apply to the converted file.
OUT-LANGUAGE	Output Content-language: value.
OUT-MESSAGE-CONTEXT	(New in 6.0) Set the output Message-Context: value
OUT-PARAMETER-NAME- <i>n</i>	Output MIME Content-Type parameter name; <i>n</i> = 0, 1, 2, ...
OUT-PARAMETER-VALUE- <i>n</i>	Output MIME Content-Type parameter value corresponding to OUT-PARAMETER-NAME- <i>n</i> .
OUT-SUBTYPE	Output MIME type if it is different than the input MIME type.
OUT-TYPE	Output MIME type if it is different than the input type
OVERRIDE-HEADER-FILE	0 or 1. If set, then MIME headers are read from the OUTPUT_HEADERS symbol, overriding the original MIME headers in the enclosing part.
OVERRIDE-OPTION-FILE	0 or 1. If set, then the conversion channel reads options from the OUTPUT_OPTIONS symbol.
PARAMETER-COPY- <i>n</i>	A list of the Content-Type: parameters to copy from the input body part's Content-Type: parameter list to the output body part's Content-Type: 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>m</i> clause. Wildcards may be used in the argument. In particular, an argument of * means to copy all the original Content-Type: parameters.

¹ Except see the RELABEL, SERVICE-CALL, and SERVICE-COMMAND parameters, which cause entries to be ignored during conversion channel processing, but do affect character set conversion.

² See [Service conversions](#) for more information on character set conversion and using the SERVICE-COMMAND parameter.

³ See [Relabelling MIME header lines](#) for more information on character set conversion and using the RELABEL parameter.

38.1.3.3 Conversion entry parameter value wildcard matching

The values of conversion entry parameter values may be specified as literal strings, or using wildcards as in MTA [mapping table entry patterns](#).

For instance,

`in-dparameter-name-0=filename; in-dparameter-value-0=*.wpc;`

would match any Content-disposition: header filename parameter that has a .wpc extension.
Or

`in-dparameter-name-0=filename; in-dparameter-value-0=*.wp${cd56}%;`

would match any Content-disposition: header filename parameter that has a .wpc, .wpd, .wp5, or .wp6 extension.

38.1.3.4 Conversion predefined symbols and environment variables

[Symbols in conversion file entries](#) shows symbols available for use within conversion file entries; that is, symbol substitutions available directly in the conversion file. See Symbol substitution in conversion entries for discussion of how to make use of such symbols in a conversion entry. (In contrast, [Environment variables for use by conversion channel shell procedures](#) below shows environment variables available for use *within a conversion shell script procedure*.)

Table 38.3 Symbols in conversion file entries

Symbol	Description
A1-FORMAT	
A1-FUNCTION	
A1-TYPE	
DESCRIPTION	The content description of the input message part.
DISPOSITION	The content disposition of the input message part.
LANGUAGE	The language tag from the Content-language: header line of the input message part.
SERVICE	
SUBJECT	The Subject: of the enclosing message/rfc822 part.
TAG	The conversion tag(s) of the input message.

Symbols may be substituted into a conversion entry by enclosing the symbol name in single quotes.

To obtain a literal single quote in a conversion entry, quote it with the backslash character, \ '.
To obtain a literal backslash in a conversion entry, use two backslashes, \ \.

[Environment variables for use by conversion channel shell procedures](#) shows the basic set of environment variables (UNIX) available for use by the conversion command.

Table 38.4 Environment variables for use by conversion channel shell procedures

Environment variable	Description
ATTACHMENT_NUMBER	The sequential number of the part, starting at number 0 for the first part of the message. Compare with PART_NUMBER, which is the MIME structured number for the part.

CONVERSION_TAG	The conversion tag(s) of the input message (only defined if the message has one or more conversion tags set).
INPUT_CHANNEL	The source channel.
INPUT_ENCODING	The encoding originally present on the body part.
INPUT_FILE	The name of the file containing the original body part. The converter should read this file.
INPUT_HEADERS	The name of the file containing the original headers for the enclosing part. The converter should read this file.
INPUT_DESCRIPTION	The content description of the input message part.
INPUT_DISPOSITION	The content disposition of the input message part.
INPUT_LANGUAGE	The content language of the input message part.
INPUT_TYPE	The content type of the input message part.
INPUT_SUBTYPE	The content subtype of the input message part.
MESSAGE_HEADERS	The name of the file containing the original headers for an enclosing message. The converter should read this file.
OUTPUT_CHANNEL	The destination channel.
OUTPUT_FILE	The name of the file where the converter should store its output. The converter should create and write this file.
OUTPUT_HEADERS	The name of the file where the converter should store MIME headers for an enclosing part. The converter should create and write this file. Note that the file should have a format of header line, header line,..., blank line; be sure to include the final blank line.
OUTPUT_OPTIONS	The name of the file from which the converter should read options (such as status values on UNIX).
PART_NUMBER	The MIME structured number for the part. Compare with ATTACHMENT_NUMBER, which is the sequential number of the part.
PART_SIZE	The size, in bytes, of the part.
<i>content-disposition parameters</i>	
<i>content-type parameters</i>	

Additional environment variables (UNIX or NT) containing Content-Type: parameter information or Content-disposition: parameter information can be created as they are needed using the PARAMETER-SYMBOL-n and DPARAMETER-SYMBOL-n facilities, respectively; see [Available conversion parameters, grouped by functionality](#).

[Override options for passing information back to the conversion channel](#) shows additional "override" options available (on UNIX) for use by the conversion channel. The converter procedure may use these to pass information back to the conversion channel. To set these options on UNIX, set OVERRIDE-OPTION-FILE=1 in the desired conversion entry and then have the converter procedure set the desired options in the OUTPUT_OPTIONS file.

Table 38.5 Override options for passing information back to the conversion channel

Override option	Description
-----------------	-------------

OUTPUT_TYPE	The content type of the output message part.
OUTPUT_SUBTYPE	The content subtype of the output message part.
OUTPUT_DESCRIPTION	The content description of the output message part.
OUTPUT_DIAGNOSTIC	Text to include in the error text returned to the message sender if a message is forcibly bounced (via PMDF_FORCERETURN) by the conversion channel.
OUTPUT_DISPOSITION	The content disposition of the output message part.
OUTPUT_ENCODING	The content transfer encoding to use on the output message part.
OUTPUT_LANGUAGE	The content language of the output message part.
OUTPUT_MODE	The 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	The MTA exit status for the converter.

38.1.3.5 Conversion entry mapping table callouts

The value for a conversion parameter may be obtained by calling out to a mapping table. The syntax for calling out to a mapping table is

```
" 'mapping-table-name:mapping-input' "
```

For instance, with a mapping table

```
X-ATT-NAMES
```

```
postscript          PS.PS$Y
wordperfect5.1      WPC.WPC$Y
msword              DOC.DOC$Y
```

then on UNIX, a conversion entry such as the following 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"
```

38.1.3.6 Conversion script header access

When performing conversions on a message part, the conversion channel has general read access (and modification access limited to specific MIME header lines) to the headers in an enclosing part, an enclosing message/rfc822 part, or to the outermost message headers if there is no enclosing message/rfc822 part. (For more general modifications of message header lines, beyond the attachment labelling modifications available to the conversion channel, see instead the [Sieve editheader extension](#).)

For instance, the IN-A1-TYPE and IN-A1-FORMAT parameters can be used to check the A1-Type: and A1-Format: headers of an enclosing part, and the OUT-A1-TYPE and OUT-A1-FORMAT parameters can be used to set those enclosing headers. Or decisions about interior message part processing can be made based upon the message's outermost headers.

More generally, if an entry is selected that has ORIGINAL-HEADER-FILE=1, then the headers of that part are written to the file represented by the INPUT_HEADERS symbol. If an entry is selected that has MESSAGE-HEADER-FILE=1, then all the original headers of the enclosing message/rfc822 part are written to the file represented by the MESSAGE_HEADERS symbol. Or if an entry is selected that has MESSAGE-HEADER-FILE=2, then all the original headers of the outermost message are written to the file represented by the MESSAGE_HEADERS symbol.

If OVERRIDE-HEADER-FILE=1, then the conversion channel will read and use as the MIME headers on that enclosing part the contents of the file represented by the OUTPUT_HEADERS symbol. Currently, the supported MIME headers that may be set in this fashion include Content-type:, Content-disposition:, Content-transfer-encoding:, Content-mode:, Content-id:, Content-description:, Content-language:, Content-annotation:, and Content-comments:.

38.1.3.7 Conversion script exit statuses

The exit status returned by a conversion script (returned via the STATUS option in the OUTPUT_OPTIONS file on UNIX) can be used to tell the conversion channel to take one of a variety of actions. The conversion status values are defined in the file `SERVERROOT/include/pmdf_err.h`; see that file for the definitive numeric values of the status codes. Available status names, numeric values (as of this writing---but see the `pmdf_err.h` file for definitive values), and their meaning, are shown in [Conversion exit status](#).

Table 38.6 Conversion exit statuses

Name	Value	Effect
PMDF__FORCERETURN	0x0A9C857A	Force return (bounce) of original message
PMDF__FORCERETURN +1	0x0A9C857B	Force return (bounce) of message, including only a sample (lines_to_return MTA option) of the original message, or a sample (lines_to_return) of whatever contents you specify in OUTPUT_FILE
PMDF__FORCEPOST	0x0A9C8612	Force message to be redirected to the postmaster (instead of going to the original recipient(s))
PMDF__FORCEASIS	0x0A9C8632	Force message to continue on unchanged
PMDF__FORCEDELETE	0x0A9C8662	Force deletion of this part (the currently being processed part) of the message
PMDF__FORCEHOLD	0xA9C86AA	Force message to be sidelined as a .HELD message file
PMDF__FORCEDISCARD	0x0A9C86B3	Force discard of entire message
PMDF__FORCEJETTISON	0x0A9C86E3	Force message to be jettisoned (non-overridable discard)

38.2 Conversion tags

The MTA has a private mechanism of *conversion tags*, which may be set and used in a variety of ways and for a variety of purposes; besides the original use for triggering user-specific automatic documentation conversion, another common use is for causing or influencing special routing of messages.

Note that conversion tags are stored in a private-to-the-MTA field in the message envelope; they are not visible in received messages. (However, message conversion tags present on message transitting the MTA will be included in MTA message transaction log entries if the [log_conversion_tag](#) MTA option is enabled.)

Conversion tags may be added via channel options, ([sourceconversiontag](#), [destinationconversiontag](#), [deliveryflags](#)), via LDAP attributes ([ldap_source_conversion_tag](#), [ldap_conversion_tag](#), [ldap_domain_attr_source_conversion_tag](#), [ldap_domain_attr_conversion_tag](#)), via an alias option ([alias_conversion_tag](#)) or [\[CONVERSION_TAG\] alias file named parameter](#), via [recipient access mapping tables](#) or the [FROM_ACCESS](#) mapping table, or via the [Sieve conversiontag extensions](#). Conversion tags present on a message may influence certain mapping table operations ([CONVERSIONS mapping table](#), [CHARSET-CONVERSION mapping table](#), [MESSAGE-SAVE-COPY mapping table](#) when selected via [message_save_copy_flags](#), as well as various address mapping tables as controlled by [include_conversiontag](#)), and control matching and hence application of specific conversions entries. Complex effects due to, and manipulations of, conversion tags are also possible via [Sieve conversiontag extensions](#).

38.3 Character set conversion and message reformatting

One very basic [mapping table](#) in the MTA 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 reformattings should be performed.

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 with enclaves of local users accustomed to using older charsets such as ISO-2022-JP (for Japanese) or KOI8-R (for Russian) may wish to convert messages outbound to the Internet into the general UTF-8 charset, to increase interoperability with remote Internet correspondents.

38.3.1 CHARSET-CONVERSION mapping table

The [CHARSET-CONVERSION](#) mapping table specifies what sorts of channel-to-channel character set conversions and message reformatting should be done. As suggested by the mapping name, character set conversion is its primary purpose.

The [CHARSET-CONVERSION](#) mapping 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 (or at least process) the message and if so, what formatting options should be used. (If no reformatting is specified, then the MTA does not bother to check for specific character set conversions.) The input string for this first probe has (by default) 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). New in JES MS 6.3, setting bit 0 (value 1) of the `include_conversiontag` MTA option will cause this first probe to instead have the form

```
IN-CHAN=in-channel;OUT-CHAN=out-channel;TAG=tag-list;CONVERT
```

where `tag-list` is a comma-separated list of any [conversion tags](#) present on the message.

If the probe matches the pattern (left hand side) of a CHARSET-CONVERSION mapping table entry, then the resulting string (right hand side of the mapping entry) should be a comma-separated list of keywords. The following keywords are provided:

Table 38.7 CHARSET-CONVERSION mapping keywords

Keyword	Action
Always	Enable conversion
Appledouble	Convert other MacMIME formats to Appledouble format
Applesingle	Convert other MacMIME formats to Applesingle format
BASE64	Switch MIME encodings to BASE64
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
Pathworks	Convert message to Pathworks Mail format
QUOTED-PRINTABLE	Switch MIME encodings to QUOTED-PRINTABLE
Record,Text	Line wrap text/plain parts at 80 characters
Record,Text= <i>n</i>	Line wrap text/plain parts at <i>n</i> characters
RFC1154	Convert message to RFC 1154 format
Thurman	Convert some non-standard "attachments" to MIME 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

If a match is found, the MTA will perform any requested message reformatting, discussed further in [Message reformatting](#), and for text parts, also check whether charset conversion is desired, as discussed further in [Character set conversion](#). A No is assumed if no match occurs.

If the message has a [conversion tag](#) set, note that the "T" flag will be set, and this can be tested for (when a match on the template, *i.e.*, left hand side, occurred) using a \$: T test in the output string. Such tests are more commonly used in the [CONVERSIONS mapping table](#) (see Section 6.2.1, below), but under less common conditions may potentially be useful here in the CHARSET-CONVERSION mapping table.

38.3.1.1 Character set conversion

If the MTA's initial probe of the [CHARSET-CONVERSION mapping table](#) (to determine whether or not *any* character set conversion or message reformatting need be performed) 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. (Note that the [include_conversiontag](#) MTA option, regardless of setting, has no effect on the form of this second probe of the CHARSET-CONVERSION mapping table.) If no match occurs for this second probe, no character set conversion is performed (although message reformatting, *e.g.*, changes to MIME structure, may be performed in accordance with the keyword matched on the first probe). If a match does occur it should typically produce a string of the form:

```
OUT-CHARSET=out-char-set
```

Here the *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 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. Encoded-words in message headers (text encoded according to the rules of [RFC 2047](#)) will also have the specified charset conversion applied.

In addition, the following other types of output request are supported.

When working on text parts of messages, one may also specify an encoding in which the MTA should output that part:

```
OUT-ENCODING=encoding-name
```

Here *encoding-name* must be the name of an encoding supported by the MTA, namely one of (as of this writing): NONE, 8BIT, 7BIT, ATOB, BASE32, BASE64, BASE85, BINARY, BINARY-8BIT, BINHEX, BTOA, COMPRESSED-BASE64, COMPRESSED-BINARY, COMPRESSED-UUCODE, COMPRESSED-UUDECODE, COMPRESSED-UUENCODE, DEFLATE-

BASE64, DEFLATE-BINARY, DEFLATE-UUCODE, DEFLATE-UUDECODING, DEFLATE-UUENCODE, HEXADECIMAL, OLD-BASE64, PATHWORDS, QUOTED-PRINTABLE, UUCODE, UUDECODING, UUENCODE, X-ATOB, X-BASE32, X-BASE85, X-BINHEX, X-BTOA, X-C-DATA, X-COMPRESSED-UUCODE, X-COMPRESSED-UUDECODING, X-COMPRESSED-UUENCODE, X-DEFLATE-UUCODE, X-DEFLATE-UUDECODING, X-DEFLATE-UUENCODE, X-HEXADECIMAL, X-OLD-BASE64, X-PATHWORDS, X-UUCODE, X-UUDECODING, X-UUENCODE. Both an output charset and an output encoding may be specified, by separating the clauses with a comma.

For encoded-words in message header lines (material encoded using the RFC 2047 encoding rules), the OUT-ENCODING must be one of QUOTED-PRINTABLE, HEXADECIMAL, X-HEXADECIMAL, or BASE64; attempting to set any other output encoding will result in the "unknown" encoding being used.

There are also several additional options that can be applied for conversion of the charset in message headers. Specifying

OUT-CHARSET=*out-charset*, RELABEL-ONLY=1

in the template (right hand side) of a mapping entry means that the MTA will simply use the specified charset name *out-charset* wherever the *in-charset* name had appeared. That is, this is intended to be used in cases where the original charset label was wrong, and it is desired to simply override the original labelling with correct labelling (but no actual charset conversion need be performed). Specifying

IN-CHARSET=*

in the template (right hand side) of a mapping entry requests that the MTA attempt to sniff the data to attempt to determine what character set was truly used. Currently, the only useful such determination that can be made by the MTA is between US-ASCII, EUC-JP, SHIFT-JIS, and ISO-2022-JP.

Specifying

OUT-LANGUAGE=*lang-tag*

in the template (right hand side) of a mapping entry tells the MTA to set the specified language tag as the value of the Content-language: header line. Specifying

OUT-LANGUAGE=**lang-tag*

tells the MTA to insert the specified language tag with the charset name inside encoded-words on header lines, if no explicit language tag was already present in the encoded-words.

38.3.1.1.1 Converting ISO-2022-JP to UTF-8 and back

Suppose that ISO-2022-JP is used locally, but that a site wishes to convert to UTF-8 for use on the Internet. In particular, suppose at this site the channel used to send to the Internet is `tcp_local`, `tcp_lmtpcs*` channels are used to deliver to local users, and local users' submitted messages are submitted via `tcp_submit` and `tcp_auth`.

Example of Converting ISO-2022-JP to and from UTF-8

CHARSET-CONVERSION

```

! Entries selecting message traffic eligible for charset conversion
!
IN-CHAN=tcp_submit;OUT-CHAN=tcp_local;CONVERT      Yes
IN-CHAN=tcp_auth;OUT-CHAN=tcp_local;CONVERT      Yes
IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;CONVERT    Yes
!
! Disable charset conversion for all other message traffic
!
IN-CHAN=*;OUT-CHAN=*;CONVERT                      No
!
! Details of which charset to convert to which other charset
!
IN-CHAN=tcp_submit;OUT-CHAN=tcp_local;IN-CHARSET=ISO-2022-JP  \
                                                                OUT-CHARSET=UTF-8
IN-CHAN=tcp_auth;OUT-CHAN=tcp_local;IN-CHARSET=ISO-2022-JP  \
                                                                OUT-CHARSET=UTF-8
IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;IN-CHARSET=UTF-8      \
                                                                OUT-CHARSET=ISO-2022-JP

```

38.3.2 Message reformatting

As mentioned in the discussion of the [CHARSET-CONVERSION mapping table](#), that mapping can also be used to effect the conversion of attachments between MIME and several proprietary mail formats.

Examples of some of the other sorts of message reformatting which can be affected with the CHARSET-CONVERSION mapping are described in [Non-MIME binary attachment conversion](#), [Relabelling MIME header lines](#), and [MacMIME format conversions](#).

38.3.2.1 Non-MIME binary attachment conversion

Mail in certain non-standard (non-MIME) formats, *e.g.*, mail in certain proprietary Sun 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
IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;CONVERT  Yes

```

You may also wish to add entries for channels to other local mail systems. For instance, an entry for the tcp_intranet channel:

CHARSET-CONVERSION

```

IN-CHAN=tcp_local;OUT-CHAN=ims-ms;CONVERT      Yes
IN-CHAN=tcp_local;OUT-CHAN=tcp_lmtpcs*;CONVERT  Yes

```

```
IN-CHAN=tcp_local;OUT-CHAN=tcp_intranet;CONVERT    Yes
```

Alternatively, to cover every channel you can simply specify `OUT-CHAN=*`. 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 indiscriminated conversions may 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, *e.g.*, `tcp_msmail`, and use the following `CHARSET-CONVERSION` mapping table entry:

```
CHARSET-CONVERSION
```

```
IN-CHAN=*;OUT-CHAN=tcp_msmail;CONVERT            RFC1154
```

38.3.2.2 MacMIME format conversions

Macintosh files have two parts, a resource fork which contains Macintosh specific information, and a data fork which 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. (See [RFC 1740 \(MacMIME\)](#) and [RFC 1741 \(Binhex\)](#).) 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 line. 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 instance, the following `CHARSET-CONVERSION` table would tell the MTA to convert to `Appledouble` format when delivering to the Message Store via either an `ims-ms` channel or a `tcp_lmtpcs*` channel:

```
CHARSET-CONVERSION
```

```
IN-CHAN=*;OUT-CHAN=ims-ms;CONVERT                Appledouble
IN-CHAN=*;OUT-CHAN=tcp_lmtpcs*;CONVERT           Appledouble
```

The conversion to Appledouble format would only be applied to parts already in one of the MacMIME formats.

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.

38.3.2.2.1 MAC-TO-MIME-CONTENT-TYPES mapping table

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, respectively, 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 file name. For instance, 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 listed in [Sample MacMIME entries](#).

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; see [Relabelling MIME header lines](#) for a discussion of MIME relabelling.

38.3.2.2.1.1 Sample MacMIME entries

A sample MAC-TO-MIME-CONTENT-TYPES mapping table is shown.

MAC-TO-MIME-CONTENT-TYPES

```

!
! format|type|creator|filename                type/subtype
!
! Sun Sound
!   MACBINARY|554c4157|5343504c|*           AUDIO/BASIC$Y
!   BINHEX|554c4157|5343504c|*           AUDIO/BASIC$Y
! Untyped Binary Data, Output File (.out file)
!   MACBINARY|42494e41|68446d70|*         APPLICATION/OCTET-STREAM$Y
!   BINHEX|42494e41|68446d70|*         APPLICATION/OCTET-STREAM$Y
! Computer Graphics Meta
!   MACBINARY|43474d6d|474b4f4e|*         IMAGE/CGM$Y
!   BINHEX|43474d6d|474b4f4e|*         IMAGE/CGM$Y
! Microsoft Word Document, Mac Microsoft Word Document
!   MACBINARY|5744424e|4d535744|*         APPLICATION/MSWORD$Y
!   BINHEX|5744424e|4d535744|*         APPLICATION/MSWORD$Y
! Microsoft Word for Windows Template
!   MACBINARY|7344424e|4d535744|*         APPLICATION/MSWORD$Y
!   BINHEX|7344424e|4d535744|*         APPLICATION/MSWORD$Y
! Postscript
!   MACBINARY|45505346|76677264|*         APPLICATION/POSTSCRIPT$Y
!   BINHEX|45505346|76677264|*         APPLICATION/POSTSCRIPT$Y
! GIF Picture
!   MACBINARY|47494666|4a414445|*         IMAGE/GIF$Y
!   BINHEX|47494666|4a414445|*         IMAGE/GIF$Y
! HP GL/2
!   MACBINARY|4850474c|474b4f4e|*         APPLICATION/VND.HP-HPGL$Y
!   BINHEX|4850474c|474b4f4e|*         APPLICATION/VND.HP-HPGL$Y
! HyperText
!   MACBINARY|54455854|556d8136|*         TEXT/HTML$Y
!   BINHEX|54455854|556d8136|*         TEXT/HTML$Y
! IEF image
!   MACBINARY|49454620|474b4f4e|*         IMAGE/IEF$Y
!   BINHEX|49454620|474b4f4e|*         IMAGE/IEF$Y
! JFax TIFF
!   MACBINARY|54494646|4a464158|*         IMAGE/TIFF$Y
!   BINHEX|54494646|4a464158|*         IMAGE/TIFF$Y
! JPEG Picture
!   MACBINARY|4a504547|4a414445|*         IMAGE/JPEG$Y
!   BINHEX|4a504547|4a414445|*         IMAGE/JPEG$Y
! MPEG-1 IPB videostream
!   MACBINARY|4d315620|6d4d5047|*         VIDEO/MPEG$Y
!   BINHEX|4d315620|6d4d5047|*         VIDEO/MPEG$Y
! MPEG-2 IPB videostream
!   MACBINARY|4d504732|4d504732|*         VIDEO/MPEG$Y
!   BINHEX|4d504732|4d504732|*         VIDEO/MPEG$Y
! FrameMaker MIF
!   MACBINARY|54455854|4672616d|*         APPLICATION/VND.FRAMEMAKER$Y

```

Message reformatting

```
    BINHEX|54455854|4672616d|*           APPLICATION/VND.FRAMEMAKER$Y
! QuickTime Movie
    MACBINARY|4d6f6f56|74747874|*       VIDEO/QUICKTIME$Y
    BINHEX|4d6f6f56|74747874|*       VIDEO/QUICKTIME$Y
! MPEG Movie of some sort
    MACBINARY|4d504547|6d4d5047|*       VIDEO/MPEG$Y
    BINHEX|4d504547|6d4d5047|*       VIDEO/MPEG$Y
! MacWrite Document
    MACBINARY|4d573244|4d574949|*       APPLICATION/MACWRITEII$Y
    BINHEX|4d573244|4d574949|*       APPLICATION/MACWRITEII$Y
! ODA Document
    MACBINARY|4f444946|4f444120|*       APPLICATION/ODA$Y
    BINHEX|4f444946|4f444120|*       APPLICATION/ODA$Y
! Portable Document Format
    MACBINARY|50444620|4341524f|*       APPLICATION/PDF$Y
    BINHEX|50444620|4341524f|*       APPLICATION/PDF$Y
! Portable Network Graphic
    MACBINARY|504e4766|474b4f4e|*       IMAGE/PNG$Y
    BINHEX|504e4766|474b4f4e|*       IMAGE/PNG$Y
! PowerPoint Presentation
    MACBINARY|534c4433|50505433|*       APPLICATION/VND.MS-POWERPOINT$Y
    BINHEX|534c4433|50505433|*       APPLICATION/VND.MS-POWERPOINT$Y
! PostScript
    MACBINARY|54455854|76677264|*       APPLICATION/POSTSCRIPT$Y
    BINHEX|54455854|76677264|*       APPLICATION/POSTSCRIPT$Y
! Rich Text Format
    MACBINARY|54455854|4d535744|*       APPLICATION/RTF$Y
    BINHEX|54455854|4d535744|*       APPLICATION/RTF$Y
! TIFF Picture
    MACBINARY|54494646|4a565752|*       IMAGE/TIFF$Y
    BINHEX|54494646|4a565752|*       IMAGE/TIFF$Y
! Tab Separated Values
    MACBINARY|54455854|5843454c|*       TEXT/TAB-SEPARATED-VALUES$Y
    BINHEX|54455854|5843454c|*       TEXT/TAB-SEPARATED-VALUES$Y
! Mu-Law Sound
    MACBINARY|554c4157|5343504c|*       AUDIO/BASIC$Y
    BINHEX|554c4157|5343504c|*       AUDIO/BASIC$Y
! WordPerfect PC 5.1 Doc, WordPerfect PC 5.x Doc
    MACBINARY|2e575035|57504332|*       APPLICATION/WORDPERFECT5.1$Y
    BINHEX|2e575035|57504332|*       APPLICATION/WORDPERFECT5.1$Y
! Lotus Spreadsheet r2.1 (.wk1 file), Lotus Spreadsheet r1.x (.wks file)
    MACBINARY|584c424e|5843454c|*       APPLICATION/VND.LOTUS-1-2-3$Y
    BINHEX|584c424e|5843454c|*       APPLICATION/VND.LOTUS-1-2-3$Y
! WordPerfect PC 4.2 Doc
    MACBINARY|2e575034|57504332|*       APPLICATION/WORDPERFECT5.1$Y
    BINHEX|2e575034|57504332|*       APPLICATION/WORDPERFECT5.1$Y
! WordPerfect PC 6.x Doc
    MACBINARY|2e575036|57504332|*       APPLICATION/WORDPERFECT5.1$Y
    BINHEX|2e575036|57504332|*       APPLICATION/WORDPERFECT5.1$Y
! WordPerfect Graphic
    MACBINARY|57504766|474b4f4e|*       APPLICATION/WORDPERFECT5.1$Y
    BINHEX|57504766|474b4f4e|*       APPLICATION/WORDPERFECT5.1$Y
! WordPerfect Mac
```

```

MACBINARY|57504431|57504332|*      APPLICATION/WORDPERFECT5.1$Y
BINHEX|57504431|57504332|*      APPLICATION/WORDPERFECT5.1$Y
! Excel Spreadsheet
MACBINARY|584c5320|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
BINHEX|584c5320|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
! Excel Chart (.xlc file)
MACBINARY|584c4320|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
BINHEX|584c4320|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
! Excel Macro (.xlm file)
MACBINARY|584c4d20|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
BINHEX|584c4d20|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
! Excel Workspace (.xlw file)
MACBINARY|584c5720|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
BINHEX|584c5720|5843454c|*      APPLICATION/VND.MS-EXCEL$Y
! AppleSingle file
MACBINARY|42494e41|54494752|*      APPLICATION/APPLEFILE$Y
BINHEX|42494e41|54494752|*      APPLICATION/APPLEFILE$Y
! MacBinary
MACBINARY|42494e41|4d423250|*      APPLICATION/MACBINARY$Y
BINHEX|42494e41|4d423250|*      APPLICATION/MACBINARY$Y
! Gnu ZIP Archive, Gnu ZIPed Tape ARchive
MACBINARY|477a6970|477a6970|*      APPLICATION/ZIP$Y
BINHEX|477a6970|477a6970|*      APPLICATION/ZIP$Y
! BinHex
! Make sure to check the file name -- regular text type parts and UUencoded
! blobs also use these Mac-type and Mac-creator values
MACBINARY|54455854|522a6368|*.hqx  APPLICATION/MAC-BINHEX40$Y
BINHEX|54455854|522a6368|*.hqx  APPLICATION/MAC-BINHEX40$Y
! BinHexed StuffIt Archive
! Make sure to check the file name -- regular text type parts and UUencoded
! blobs also use these Mac-type and Mac-creator values
MACBINARY|54455854|522a6368|*.sithqx APPLICATION/MAC-BINHEX40$Y
BINHEX|54455854|522a6368|*.sithqx APPLICATION/MAC-BINHEX40$Y
! PGP Key File
MACBINARY|504b6579|4d504750|*      APPLICATION/PGP-KEYS$Y
BINHEX|504b6579|4d504750|*      APPLICATION/PGP-KEYS$Y
! PC ZIP Archive
MACBINARY|5a495020|5a495020|*      APPLICATION/ZIP$Y
BINHEX|5a495020|5a495020|*      APPLICATION/ZIP$Y

```

38.3.3 Relabelling MIME header lines

Some user agents or gateways may emit messages with MIME header lines which are less informative than they might be, but which nevertheless contain enough information to construct more precise MIME header lines. 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 header lines.

If the first probe of the [CHARSET-CONVERSION mapping table](#) yields a "Yes" or "Always" keyword, then the MTA will check for the existence of any conversions entries. (In Unified Configuration, the MTA checks for the existence of any [conversions](#) entries; in legacy configuration, the MTA checks for the existence of a conversions file, `IMTA_TABLE:conversions`.) If any conversions entries exist, then the MTA will look in for

an entry with RELABEL=1 and if it finds such an entry, the MTA will then perform any MIME relabellings specified in the entry.

New in 7.0.5, in addition to MIME relabellings, RELABEL=1 can also perform encoding changes (OUT-ENCODING clauses will be applied); formerly, OUT-ENCODING had no effect in RELABEL=1 entries.

For instance, the combination of a CHARSET-CONVERSION mapping table of:

```
msconfig> show mapping:CHARSET-CONVERSION
role.mapping:CHARSET-CONVERSION.rule = IN-CHAN=tcp_*;OUT-CHAN=ims-ms;CONVERT Yes
```

and conversions entries (use msconfig's edit conversions command to create such entries from within msconfig) 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-disposition=attachment;
in-dparameter-name-0=filename; in-dparameter-value-0=*.ps;
out-type=application; out-subtype=postscript;
out-disposition=attachment; dparameter-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
```

```
out-chan=ims-ms; in-type=application; in-subtype=octet-stream;
in-disposition=attachment;
in-dparameter-name-0=filename; in-dparameter-value-0=*.msw;
out-type=application; out-subtype=msword;
out-disposition=attachment; dparameter-copy-0=*; relabel=1
```

will result in messages that arrive on a tcp_* 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.)

38.3.4 Service conversions

The MTA's conversion service facility may be used to process, with site-supplied procedures, message content so as to produce a new form of the message. Unlike either the sorts of [CHARSET-CONVERSION](#) operations discussed elsewhere or the [conversion channel](#), which operate on the content of individual MIME message parts, conversion services operate on entire MIME entities (MIME headers plus content). (But note that service conversions do not give access to the outermost, non-MIME header lines.) 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.

Service conversions may be triggered using the `serviceconversion` channel option. But more typically, like other CHARSET-CONVERSION operations, conversion services are enabled through the CHARSET-CONVERSION mapping table. 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 conversions options (in Unified Configuration) or the MTA conversions file (in legacy configuration).⁶ If a conversions file exists, then the MTA will look in it for an entry specifying a SERVICE-COMMAND or SERVICE-CALL, 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
```

or

```
in-chan=channel-pattern;  
  in-type=type-pattern; in-subtype=subtype-pattern;  
  service-call=image | routine | argument
```

Of key interest is (in `service-command` entries) the `command` string. This is the command which should be executed to perform a service conversion (*e.g.*, 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 a conversions file entry 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 (UNIX) are used to pass the names of the input and output files as well as the name of a file containing some message envelope information. The names of these environment variables are:

Table 38.8 Conversion command callout environment variables on UNIX systems

Variable	Usage
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 INFO_FILE in particular contains the envelope From address, the list of the message's envelope recipient addresses, and (as of Chipotle) if present in the original message envelope the authenticated sender; this information is recorded in the form of pseudo-header lines, *e.g.*,

```
X-Author-info: authenticated-sender
X-Envelope-from: envelope-from
X-Envelope-to: recipient-list
```

The values of these three environment variables, INPUT_FILE, OUTPUT_FILE, and INFO_FILE, may be substituted into the command line by using standard command line substitution: *i.e.*, 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 entry 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
```

Note: Prior to Messaging Server 7.0, the conversions file was located via the IMTA_CONVERSION_FILE MTA Tailor file option, so usually /opt/SUNWmsgsr/config/conversions. As of Messaging Server 7.0, the conversions file is located at config-root/config/conversions.

38.4 Interactions between conversions and character set conversions

Character set conversions come into play at two points during message enqueue processing: during header processing, and during message body processing. (By header processing here we mean any actual conversions of the character set in the message header, *e.g.*, character set translation of personal names in addresses, not the MIME relabelling or other sorts of header line transformations that can also be configured via a [CHARSET-CONVERSION mapping table](#).)

After the message header processing character set conversion check and before any message body character set conversion check, the MTA consults the [CONVERSIONS mapping table](#) (if present). If a message is to be routed to the [conversion channel](#) due to a match in the CONVERSIONS mapping table, then normally character set conversion for the message body during this enqueue is disabled; it is expected instead that any relevant character set conversion of the message body will be incorporated into the conversion channel's conversion script processing of the message body parts. (The `imsimta test -translation` utility may come in useful in conversion scripts for such purposes.) Alternatively, new in JES MS 6.3, the "Preprocess" keyword may be used in channel routing entries in the CONVERSIONS mapping table to specify that character set conversion of the message body (including

RELABEL operations) should be performed during the enqueue to the conversion channel (or alternate conversion channel) rather than being disabled as normal at that stage. Or yet another alternative: character set conversion of the message body for messages coming *from* the conversion channel may be configured, to convert the character set in the message body *after* the conversion channel has performed its operations; for such configuration, note that the CHARSET-CONVERSION probe will not normally show the conversion channel as the source channel, but rather show the "original" source channel as the source channel.



Chapter 39 MTA options

39.1 MTA option naming in Unified Configuration	39-8
39.2 Legacy configuration MTA option file	39-9
39.2.1 Option value syntax in legacy configuration	39-9
39.3 Getting option changes to take effect on the MTA	39-10
39.4 MTA options listed alphabetically	39-11
39.5 MTA options listed by functional group	39-24
39.6 enable Option Under mta	39-54
39.7 Alias and address MTA options	39-54
39.7.1 alias_case MTA option	39-55
39.7.2 alias_domains MTA option	39-56
39.7.3 alias_magic MTA option	39-56
39.7.4 delimiter_char MTA option	39-57
39.7.5 exproute_forward MTA option	39-57
39.7.6 improute_forward MTA option	39-57
39.7.7 local_format_restrictions MTA option	39-57
39.7.8 max_alias_levels MTA option	39-58
39.7.9 missing_recipient_group_text MTA option	39-58
39.7.10 missing_recipient_policy MTA option	39-58
39.7.11 name_table_name MTA option	39-59
39.7.12 reverse_envelope MTA option	39-60
39.7.13 subaddress_char MTA option	39-60
39.7.14 token_char MTA option	39-60
39.7.15 use_alias_database MTA option	39-60
39.7.16 use_domain_database MTA option	39-61
39.7.17 use_forward_database MTA option	39-61
39.7.18 use_personal_aliases MTA option	39-62
39.7.19 use_reverse_database MTA option	39-62
39.7.20 user_case MTA option	39-64
39.8 Autoresponse periodicity MTA options	39-64
39.8.1 autoreply_timeout_default MTA option	39-65
39.8.2 notify_maximum_timeout MTA option	39-65
39.8.3 notify_minimum_timeout MTA option	39-66
39.8.4 notify_timeout_default MTA option	39-66
39.8.5 vacation_cleanup MTA option	39-66
39.8.6 vacation_maximum_timeout MTA option	39-66
39.8.7 vacation_minimum_timeout MTA option	39-66
39.8.8 vacation_template MTA option	39-67
39.9 BURL MTA options	39-68
39.9.1 imap_password MTA option	39-68
39.9.2 imap_username MTA option	39-68
39.10 Configutil override MTA options	39-68
39.11 Conversions MTA options	39-69
39.11.1 conversions MTA option	39-69
39.12 Counters MTA options	39-69
39.12.1 circuitcheck_completed_bins MTA option	39-69
39.12.2 enable_delay_timers MTA option	39-70
39.12.3 log_delay_bins MTA option	39-70
39.12.4 log_frustration_limit MTA option	39-70
39.12.5 log_size_bins MTA option	39-70
39.12.6 log_statistics MTA option	39-70

39.13 Database MTA options	39-71
39.14 Debug MTA options	39-71
39.14.1 loglevel Option	39-72
39.14.2 ap_debug MTA option	39-73
39.14.3 cache_debug MTA option	39-73
39.14.4 config_debug MTA option	39-73
39.14.5 debug_flush MTA option	39-73
39.14.6 dequeue_debug MTA option	39-73
39.14.7 filter_debug MTA option	39-74
39.14.8 log_debug MTA option	39-74
39.14.9 mm_debug MTA option	39-74
39.14.10 os_debug MTA option	39-74
39.14.11 post_debug MTA option	39-75
39.14.12 return_debug MTA option	39-75
39.14.13 return_verify Option	39-75
39.14.14 symbiont_debug Option	39-75
39.14.15 tracking_debug MTA option	39-75
39.15 Direct LDAP MTA options	39-76
39.15.1 LDAP bind and connect MTA options	39-76
39.15.2 Direct LDAP domain lookup MTA options	39-78
39.15.3 Direct LDAP usergroup lookup MTA options	39-84
39.15.4 Direct LDAP schema MTA options	39-88
39.15.5 Direct LDAP attribute interpretation MTA options	39-91
39.15.6 Direct LDAP attribute name MTA options	39-102
39.15.7 Direct LDAP attributes returned upon authentication MTA options	39-151
39.15.8 LDAP lookup cache MTA options	39-152
39.16 Directory location MTA options	39-154
39.16.1 tmpdir MTA option	39-154
39.16.2 langdir MTA option	39-154
39.17 DKIM MTA options	39-154
39.17.1 dkim_ignore_domains MTA option	39-154
39.17.2 dkim_preserve_domains MTA option	39-155
39.17.3 dkim_remove_domains MTA option	39-155
39.18 DNS lookup MTA options	39-155
39.18.1 blocked_mail_from_ips MTA option	39-155
39.18.2 return_envelope MTA option	39-155
39.19 Error text and error interpretation MTA options	39-156
39.19.1 access_errors MTA option	39-156
39.19.2 error_text MTA options	39-157
39.19.3 use_permanent_error MTA option	39-168
39.19.4 use_temporary_error MTA option	39-169
39.20 External filtering context MTA options	39-170
39.20.1 scan_schannel MTA option	39-170
39.20.2 scan_originator MTA option	39-170
39.20.3 scan_recipient MTA option	39-170
39.21 File format MTA options	39-170
39.21.1 buffer_size MTA option	39-171
39.21.2 cache_magic MTA option	39-171
39.21.3 cbt MTA option	39-171
39.21.4 comment_chars MTA option	39-171
39.21.5 debug_flush MTA option	39-172
39.21.6 dequeue_map MTA option	39-172
39.21.7 fdirectory MTA option	39-172

39.21.8	fsync MTA option	39-172
39.21.9	log_alq MTA option	39-173
39.21.10	log_deq MTA option	39-173
39.21.11	max_internal_blocks MTA option	39-173
39.21.12	mm_mbc MTA option	39-173
39.21.13	mm_mbf MTA option	39-173
39.21.14	notary_quote MTA option	39-174
39.21.15	osync MTA option	39-174
39.21.16	projectid Option Under mta	39-174
39.21.17	queue_cache_mode MTA option	39-174
39.21.18	queue_cache_mode_3_files MTA option	39-174
39.21.19	use_text_databases MTA option	39-174
39.22	Internal size MTA options	39-175
39.22.1	alias_hash_size MTA option	39-176
39.22.2	alias_member_size MTA option	39-176
39.22.3	channel_table_size MTA option	39-177
39.22.4	chunk_cache_limit MTA option	39-177
39.22.5	circuitcheck_paths_size MTA option	39-177
39.22.6	conversion_size MTA option	39-177
39.22.7	describe_cache_limit MTA option	39-177
39.22.8	domain_hash_size MTA option	39-178
39.22.9	file_member_size MTA option	39-178
39.22.10	forward_data_size MTA option	39-178
39.22.11	fruits_size MTA option	39-178
39.22.12	general_data_size MTA option	39-178
39.22.13	host_hash_size MTA option	39-179
39.22.14	ldap_attr_name_hash_size MTA option	39-179
39.22.15	ldap_object_class_hash_size MTA option	39-179
39.22.16	map_names_size MTA option	39-180
39.22.17	options_hash_size MTA option	39-180
39.22.18	personal_conversion_size MTA option	39-180
39.22.19	reverse_data_size MTA option	39-180
39.22.20	string_pool_size_n MTA options	39-181
39.22.21	wild_pool_size MTA option	39-181
39.23	LDAP external directory lookup MTA options	39-181
39.23.1	ldap_ext_host MTA option	39-182
39.23.2	ldap_ext_max_connections MTA option	39-182
39.23.3	ldap_ext_password MTA option	39-182
39.23.4	ldap_ext_port MTA option	39-182
39.23.5	ldap_ext_username MTA option	39-182
39.24	LDAP PAB MTA options	39-182
39.24.1	ldap_pab_host MTA option	39-183
39.24.2	ldap_pab_max_connections MTA option	39-183
39.24.3	ldap_pab_password MTA option	39-183
39.24.4	ldap_pab_port MTA option	39-183
39.24.5	ldap_pab_username MTA option	39-183
39.25	Mailing list MTA options	39-184
39.25.1	defer_group_processing MTA option	39-184
39.25.2	digest_on MTA option	39-184
39.25.3	expandable_default MTA option	39-184
39.25.4	mail_off MTA option	39-185
39.25.5	or_clauses MTA option	39-185
39.25.6	post_off MTA option	39-185

39.26	MAILSERV MTA options	39-186
39.26.1	MAILSERV moderator MTA options	39-186
39.26.2	MAILSERV LDAP schema MTA options	39-186
39.26.3	MAILSERV user LDAP attribute name MTA options	39-186
39.26.4	MAILSERV list subscription LDAP attribute name MTA options	39-187
39.26.5	MAILSERV list LDAP attribute name MTA options	39-187
39.27	Mapping table MTA options	39-188
39.27.1	Access mapping table MTA options	39-188
39.27.2	Miscellaneous mapping table MTA options	39-196
39.28	Memcache MTA options	39-202
39.28.1	memcache_host MTA option	39-202
39.28.2	memcache_port MTA option	39-202
39.28.3	memcache_expire MTA option	39-202
39.28.4	memcache_timeout MTA option	39-202
39.28.5	alias_database_url MTA option	39-202
39.28.6	domain_database_url MTA option	39-202
39.28.7	forward_database_url MTA option	39-203
39.28.8	general_database_url MTA option	39-203
39.28.9	reverse_database_url MTA option	39-203
39.28.10	ssr_database_url MTA option	39-203
39.29	Message archival and hashing MTA options	39-203
39.29.1	journal_format MTA option	39-204
39.29.2	message_hash_algorithm MTA option	39-204
39.29.3	message_hash_fields MTA option	39-204
39.29.4	unique_id_template MTA option	39-205
39.30	Message size MTA options	39-205
39.30.1	block_limit MTA option	39-205
39.30.2	block_size MTA option	39-206
39.30.3	bounce_block_limit MTA option	39-206
39.30.4	content_return_block_limit MTA option	39-207
39.30.5	header_limit MTA option	39-207
39.30.6	line_limit MTA option	39-207
39.30.7	local_quota_checks MTA option	39-207
39.30.8	max_header_block_use, max_header_line_use MTA options	39-208
39.30.9	max_header_blocks MTA option	39-208
39.30.10	max_header_lines MTA option	39-208
39.30.11	max_mime_levels, max_mime_parts MTA options	39-209
39.30.12	*_block_limit priority limit MTA options	39-209
39.31	Message tracking MTA options	39-210
39.31.1	tracking_mode MTA option	39-210
39.31.2	tracking_retries, tracking_retry_delay MTA options	39-210
39.32	MeterMaid MTA options	39-211
39.32.1	metermaid_backoff MTA option	39-211
39.32.2	metermaid_expire MTA option	39-211
39.32.3	metermaid_host MTA option	39-211
39.32.4	metermaid_port MTA option	39-211
39.32.5	metermaid_secret MTA option	39-211
39.32.6	metermaid_timeout MTA option	39-212
39.33	MLS MTA options	39-212
39.33.1	mls Option	39-212
39.34	MTQP MTA options	39-212
39.34.1	mtqp_port MTA option	39-212
39.34.2	mtqp_timeout MTA option	39-212

39.34.3	mtqp_expire MTA option	39-212
39.35	Notification message MTA options	39-213
39.35.1	bounce_block_limit MTA option	39-213
39.35.2	content_return_block_limit MTA option	39-213
39.35.3	history_to_return MTA option	39-213
39.35.4	lines_to_return MTA option	39-214
39.35.5	notary_decode MTA option	39-214
39.35.6	notary_quote MTA option	39-214
39.35.7	return_address MTA option	39-215
39.35.8	return_delivery_history MTA option	39-215
39.35.9	return_envelope MTA option	39-215
39.35.10	return_personal MTA option	39-216
39.35.11	return_units MTA option	39-216
39.35.12	use_precedence MTA option	39-217
39.35.13	use_warnings_to MTA option	39-217
39.36	Password and TLS MTA options	39-217
39.36.1	plaintextmncipher Option Under mta	39-217
39.36.2	smtpproxypassword Option	39-218
39.36.3	sslnicknames Option Under mta	39-218
39.37	Processing priority MTA options	39-218
39.37.1	mtpriority_policy MTA option	39-219
39.37.2	*_block_limit priority limit MTA options	39-219
39.38	Received header line MTA options	39-220
39.38.1	held_sndopr MTA option	39-220
39.38.2	id_domain MTA option	39-221
39.38.3	max_local_received_lines MTA option	39-221
39.38.4	max_mr_received_lines MTA option	39-221
39.38.5	max_received_lines MTA option	39-221
39.38.6	max_total_received_lines MTA option	39-222
39.38.7	max_x400_received_lines MTA option	39-222
39.38.8	received_domain MTA option	39-222
39.38.9	received_version MTA option	39-222
39.39	Sieve filter MTA options	39-223
39.39.1	systemfilter MTA option	39-223
39.39.2	Sieve filter interpretation MTA options	39-223
39.39.3	Sieve filter limit MTA options	39-225
39.39.4	Sieve filter caching MTA options	39-227
39.39.5	Sieve language extension MTA options	39-228
39.39.6	Sieve filter duplicate extension MTA options	39-229
39.39.7	Sieve filter error text MTA options	39-231
39.39.8	Sieve filter log and debug MTA options	39-231
39.40	Spamfilter MTA options	39-233
39.40.1	optin_user_carryover MTA option	39-234
39.40.2	spamfilterN_library MTA options	39-234
39.40.3	spamfilterN_config_file MTA options	39-235
39.40.4	spamfilterN_null_optin MTA options	39-235
39.40.5	spamfilterN_*_M action and verdict MTA options	39-235
39.40.6	spamfilterN_final MTA options	39-238
39.40.7	spamfilterN_includeheaders MTA options	39-238
39.40.8	spamfilterN_null_action MTA options	39-238
39.40.9	spamfilterN_received MTA options	39-238
39.40.10	spamfilterN_returnpath MTA options	39-239
39.40.11	spamfilterN_string_action MTA options	39-239

39.41	SPF MTA options	39-240
39.41.1	spf_smtp_status_fail MTA option	39-240
39.41.2	spf_smtp_status_fail_all MTA option	39-241
39.41.3	spf_smtp_status_permerror MTA option	39-241
39.41.4	spf_smtp_status_softfail MTA option	39-242
39.41.5	spf_smtp_status_softfail_all MTA option	39-242
39.41.6	spf_smtp_status_temperror MTA option	39-243
39.41.7	spf_max_dns_queries MTA option	39-244
39.41.8	spf_max_recursion MTA option	39-244
39.41.9	spf_max_time MTA option	39-244
39.42	SRS MTA options	39-244
39.42.1	srs_domain, srs_maxage, srs_secrets MTA Options	39-245
39.42.2	token_char MTA option	39-246
39.43	Syslog MTA options	39-246
39.43.1	held_sndopr MTA option	39-246
39.43.2	log_connections_syslog MTA option	39-247
39.43.3	log_messages_syslog MTA option	39-247
39.43.4	log_sndopr MTA option	39-249
39.43.5	sndopr_priority MTA option	39-249
39.43.6	spamfilterN_optional MTA options	39-250
39.44	Transaction logging MTA options	39-251
39.44.1	log_alternate_recipient MTA option	39-252
39.44.2	log_auth MTA option	39-252
39.44.3	log_callout_delays MTA option	39-252
39.44.4	log_connection MTA option	39-254
39.44.5	log_conversion_tag MTA option	39-255
39.44.6	log_deliver_by MTA option	39-255
39.44.7	log_diagnostics MTA option	39-256
39.44.8	log_envelope_id MTA option	39-256
39.44.9	log_filename MTA option	39-256
39.44.10	log_filter MTA option	39-256
39.44.11	log_format MTA option	39-257
39.44.12	log_futurerelease MTA option	39-261
39.44.13	log_header MTA option	39-261
39.44.14	log_imap_flags MTA option	39-262
39.44.15	log_delivery_flags MTA option	39-262
39.44.16	log_intermediate MTA option	39-262
39.44.17	log_local MTA option	39-263
39.44.18	log_mailbox_uid MTA option	39-263
39.44.19	log_message_id MTA option	39-263
39.44.20	log_mtpriority MTA option	39-264
39.44.21	log_node MTA option	39-264
39.44.22	log_notary MTA option	39-265
39.44.23	log_priority MTA option	39-265
39.44.24	log_process MTA option	39-266
39.44.25	log_queue_time MTA option	39-266
39.44.26	log_reason MTA option	39-267
39.44.27	log_sensitivity MTA option	39-267
39.44.28	log_times MTA option	39-268
39.44.29	log_tracking MTA option	39-268
39.44.30	log_transactionlog MTA option	39-268
39.44.31	log_uid MTA option	39-269
39.44.32	log_use_xtext MTA option	39-269

39.44.33	<code>log_username</code> MTA option	39-269
39.44.34	<code>separate_connection_log</code> MTA option	39-270
39.44.35	<code>return_split_period</code> MTA option	39-270
39.44.36	<code>return_cleanup_period</code> MTA option	39-271
39.45	OpenVMS user agent MTA options	39-271
39.45.1	<code>delivery_receipt_off</code> Option	39-271
39.45.2	<code>delivery_receipt_on</code> Option	39-271
39.45.3	<code>dis_nesting</code> MTA option	39-271
39.45.4	<code>form_names</code> MTA option	39-272
39.45.5	<code>mail_delivery_filename</code> Option	39-272
39.45.6	<code>missing_address</code> Option	39-272
39.45.7	<code>multinet_mm_exclusive</code> Option	39-272
39.45.8	<code>read_receipt_off</code> MTA option	39-272
39.45.9	<code>read_receipt_on</code> MTA option	39-272
39.45.10	<code>safe_tcl_mode</code> MTA option	39-273
39.45.11	<code>use_mail_delivery</code> Option	39-273
39.45.12	<code>vms_mail_exclusive</code> Option	39-273

A subset of the overall Messaging Server configuration options are the MTA options.

The `mta.enable` option enables startup of the MTA upon `start-msg` startup.

The MTA uses options to provide a means of overriding the default values of various parameters that apply to the MTA as a whole. Various MTA [channels](#) also have their own channel-level options. The general MTA options have the same format as MTA [channel options](#) but are otherwise distinct --- they apply to the MTA as a whole rather than being restricted in application to any specific channel.

In older versions of the MTA, such options were stored in the so-called *MTA option file*, normally named `option.dat`. So the MTA options are also sometimes referred to as "option.dat options".

A variety of configuration options are controlled by MTA options. In particular, some MTA options establish sizes of the various internal, in-memory tables into which the remainder of the configuration --- the channel definitions, mapping tables, conversion entries, *etc.* --- are read.

Additional discussions of MTA options include:

- [Legacy configuration MTA option file](#)
- [Getting option changes to take effect on the MTA](#)
- [Option value syntax in legacy configuration](#)
- [MTA options, listed alphabetically](#)
- [MTA options, listed by functional group](#)

See also the `umask` Message Store option, which affects more than only Message Store files.

In contrast to MTA options, which typically modify fundamental aspects of overall MTA operation, another important category of option modifying MTA operation more specifically at the channel level is that of [channel options](#). Note that MTA options and channel options are *mostly* distinct, as they mostly modify qualitatively different aspects of operation; however, there are some cases where an MTA option establishes a general default for all channels, which a channel option may then override on a per-channel basis.

39.1 MTA option naming in Unified Configuration

In the Unified Configuration, MTA configuration including MTA options (meaning here MTA-as-a-whole options, as opposed to channel-specific options) are set in the `config.xml` Messaging Server unified configuration file where they are XML elements. However, under normal circumstances, the Messaging Server unified configuration file `config.xml` is not--- indeed should not be--- inspected or edited manually by the MTA administrator. Instead, normally the MTA's `msconfig` utility is used to examine the MTA configuration and make configuration changes.

The `msconfig` utility presents a command line interface for viewing and editing MTA settings corresponding to the underlying XML constructs from `config.xml`; it is the recommended interface for examining the MTA configuration and making MTA configuration changes. The `msconfig` utility represents MTA options in the general form:

```
instance-or-role.mta.option-name
```

That is, in the `msconfig` representation, MTA options are either of the form:

```
instance.mta.option-name
```

or

```
role.mta.option-name
```

However, the `instance.` and `role.` prefixes are *usually* omitted, since the `msconfig` utility is typically used in modes in which the appropriate instance or role is determined automatically by the utility, with the instance or role scope then set appropriately automatically by the utility. That is, usual usage in `msconfig` is to refer simply to:

```
mta.option-name
```

Furthermore the `mta.` prefix is often unnecessary. If the MTA option name is unambiguous (does not collide with any option name for any other component of Messaging Server), then the MTA scope (`mta.` prefix) will also be automatically supplied by the `msconfig` utility, so that often simply referring to:

```
option-name
```

will suffice. However, especially in any scripting that may be reused later, it may be best to include the `mta.` prefix to avoid the potential for future option additions to result in a name ambiguity.

Thus in typical use, MTA options (under the appropriate instance or role) would typically be shown, set, or unset using the `msconfig` commands, respectively,

```
msconfig> show mta.option-name  
msconfig> set mta.option-name  
msconfig> unset mta.option-name
```

The `msconfig` utility will translate between this simple `mta.option-name` representation and the (more complex form of the) underlying XML elements stored in `config.xml`.

See the `msconfig` utility for more details.

39.2 Legacy configuration MTA option file

On UNIX and NT systems, the MTA option file is the file `.../config/option.dat`.

Each time an MTA program begins running in legacy configuration mode, the MTA option file is read and loaded into memory. This overhead may be avoided by compiling your MTA configuration, in which case the contents of the option file will be incorporated into the compiled configuration. The disadvantage to this, however, is that it means that the configuration must be recompiled and reinstalled whenever a change is made to the option file. See [Compiling the MTA configuration](#) for details on compiling your configuration.

The MTA option file is line-oriented, where each line contains the setting for one option. An option setting has the form:

```
option=value
```

where note **white space is significant**, and white space should *not* be present before the equal sign. (Indeed, for string values, even **trailing white space** is significant and -- normally -- preserved as it is meaningful and desired for some options, though as of Messaging Server 7.0, the MTA will attempt some "clean up" of certain option values, such as those for setting file names, and such "clean up" can potentially include removal of extraneous trailing spaces from the option value setting.)

value may be either a string, an integer, a floating point value, or a comma-separated list of one such type of value, depending on the option's requirements. For further details on value syntax, see [Option value syntax in legacy configuration](#).

Long lines may be continued by ending them with a backslash, `\`.

Comments are allowed. Any line that begins in column one with an exclamation point, semicolon, or sharp character, `!`, `;`, or `#`, is considered to be a comment and is ignored (even if the preceding line ended with a backslash, which would normally mean that the line was a continuation). For the MTA option file, the interpretation of the characters `!`, `;`, and `#` as indicating a comment line is hard-coded; in particular, it is not affected by any setting of the `comment_chars` MTA option (though that option does affect the interpretation of *other* MTA configuration files, including other "option files").

Whereas white space on an option setting line is significant, note that blank lines are permitted and ignored in any option file.

Note: This same format is used for various additional MTA "option files", in addition to "the" MTA option file. That is, some spam/virus filter package option files, and in legacy configuration the Dispatcher option file, Job Controller option file, channel option files, *etc.*, all use this same sort of format.

39.2.1 Option value syntax in legacy configuration

Option values are typically one of string, integer, or (less commonly) floating point, or in some cases comma-separated lists of one of the above types of values, or in other cases space-separated lists of one of the above types of values. Furthermore, some option values are further constrained, as for instance an integer in a particular range, or a string that corresponds to an

LDAP attribute name or to a channel name; and some option values have additional semantics as for instance a bit-encoded integer, or a boolean integer. The type of value valid for an option depends upon the option's requirements; the exact type required will be described for each option. The discussion below will touch on general syntax issues.

The length of the string specifying the option value (the length of the material to the right of the equal sign) is limited to 1024 characters; (prior to JES MS 6.2p8/6.3, the limit had been 256 characters). For options where the value involves substitution expansions (such as [URL values](#) using [URL substitution sequences](#)), the limit *after* such expansion is (and has been) 1024 characters.

If an 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 .

Bit-encoded integer values are used for a number of options. A bit-encoded integer value may be written as any integer would be; however, it has additional semantics. In a bit-encoded integer value, each bit of the integer controls a different feature; thus a bit-encoded integer value expresses a whole set of enable/disable choices. The least significant bit is conventionally referred to a "bit 0". The integer value of each set bit is then two raised to the bit position power; for instance, setting bit 0 means adding 1 to the value, setting bit 1 means adding 2 to the value, setting bit 2 means adding 4 to the value, *etc.*

Boolean values may be specified either as a number, 0 (false) or 1 (true), or as a string "false" or "true".

Floating point values may be written either in usual decimal notation, for instance, 3.1416, or may be written using the common scientific exponential notation, for instance, 1.045E2 meaning 104.5. The exponent indicator may be any of the characters E, e, F, or f.

For string values, note that spaces are significant in values, **including trailing spaces**, as for certain options values with trailing spaces are meaningful. (However, as of Messaging Server 7.0, the MTA will attempt some "clean up" of certain option values, such as those for setting file names, or for reading options from the [Dispatcher](#) or [Job Controller](#) configuration files -- where such "clean up" can potentially include removal of extraneous trailing spaces from the option value setting.)

39.3 Getting option changes to take effect on the MTA

Options, both [MTA options](#) and [channel options](#), are part of the base MTA configuration. When an MTA process starts up, one of its first tasks is to get initialized with a complete view of the current "live" base MTA configuration. When a compiled configuration is in use (exists), then the current "live" base MTA configuration is the most recently compiled version of the MTA configuration; any options with values updated subsequent to the most recent compilation are not "live" (will not have an effect). When a compiled configuration is not in use, then any changes to option values are immediately "live" (though this does not mean that they will take immediate effect; see below).

Now since many MTA processes are long-running, and since they normally initialize with the MTA configuration information upon first starting up, even "live" configuration changes are not seen immediately by already running MTA processes. To update already running MTA processes with new configuration information, the [reload utility](#) (or one of the utilities that subsumes it) must be used. Note that since MTA processes do automatically "cycle" --- that is, time out and expire to be replaced with new processes --- any changes to the "live"

configuration will get seen eventually, if not immediately, even if the [reload utility](#) is not used.

So in particular, when a compiled configuration is being used, then options are part of the compiled configuration. And in order to get changes to options to take effect, you must: (1) re-compile the configuration (using the [cnbuild utility](#) or a utility that subsumes it), and furthermore for already running, long-running processes you must also (2) reload the configuration (using the [reload utility](#) or a utility that subsumes it).

See also: [Updating the configuration](#), [Reloading the configuration](#), [refresh utility](#), [Long-running MTA processes](#), [Restarting the MTA](#).

39.4 MTA options listed alphabetically

[Table of MTA options, listed alphabetically](#), shown below, lists the available options in alphabetic order. A brief description of each option is included, though more details can be found in the discussions of the individual options. See also [MTA options, listed by functionality](#), which lists the options grouped together by function, and which furthermore includes links to subsections with general information about the groups of options.

Table 39.1 MTA options, listed alphabetically

Option	Usage
access_auth	(New in 8.0) Control whether FROM_ACCESS mapping table probes include the SMTP MAIL FROM AUTH parameter's value and/or the canonical username produced by authentication
access_counts	Control whether recipient access mapping table probes include a recipient count field
access_errors	Control the information issued in certain error messages
access_orcpt	Control whether recipient access mapping table probes include an ORCPT field
alias_case	Control the case sensitivity of aliases; support for this option is RESTRICTED: set it only if instructed to do so by Oracle engineering
alias_domains	Control the format of alias file (and hence Unified Configuration alias options) and alias database lookups
alias_entry_cache_negative	Whether to cache negative results (<i>i.e.</i> , failures) of alias lookups (alias_urlN lookups)
alias_entry_cache_size	Number of alias lookup (alias_urlN) results to keep cached
alias_entry_cache_timeout	How long to cache alias lookup (alias_urlN) results
alias_magic	Control the order in which alias lookups are performed; support for this option is RESTRICTED
alias_url0	URL for doing alias lookups
alias_url1	URL for doing alias lookups

MTA options listed alphabetically

alias_url2	URL for doing alias lookups
alias_url3	URL for doing alias lookups
alias_hash_size	Set the number of aliases allowed in the alias file
alias_member_size	Set the number of alias expansions allowed: in the alias file (legacy configuration), or via alias option (Unified Configuration)
aliasdetourhost_null_optin	(New in JES MS 6.2p4) Specify a value for the attribute named by the ldap_detourhost_optin MTA option (and new in MS 7.0.5, also the attribute named by the ldap_domain_attr_detourhostoptin) that means that the attribute should be ignored; the purpose is to allow a way of "turning off" message detouring (where the detouring is typically for purposes of spam/virus filtering) for sites and users whose provisioning facilities find it easier to set an attribute value rather than to remove an attribute entirely
allow_unquoted_addrs_violate_rfc2798	Look up addresses in LDAP with quotes stripped from the local part
ap_debug	Internal MTA debugging option: enable debugging of low level address parsing; support for this option is RESTRICTED
autoreply_timeout_default	Specify a default duration between successive vacation responses to the same sender
averages_cache_size	(New in JES MS 6.2) Size of the cache of load average values
averages_cache_timeout	(New in JES MS 6.2) Timeout for the cache of load average values
block_limit	Limit the size of messages allowed through the MTA
block_size	Set the size of MTA "blocks"
blocked_mail_from_ips	Specify IP address literals whose A records should be ignored for purposes of the lookups done due to the mailfromdnsverify channel keyword (new in 6.1-0.01 release)
bounce_block_limit	Limit the amount of original message content included in bounce messages
brightmail_*	Synonyms for the corresponding spamfilter_* options ; see the spamfilter_* options for definitions
buffer_size	Set the buffer size used when writing message files; default is 8192
cache_debug	Output direct LDAP lookup caching statistics
cache_magic	OBSOLETE: Control the sorting (for processing purposes) of old message files

<code>channel_table_size</code>	Set the number of channels allowed in the MTA configuration
<code>circuitcheck_completed_bins</code>	Specify the bins for the message circuit check message counters
<code>circuitcheck_paths_size</code>	Number of circuit check paths (entries) to allow in the circuit check configuration file
<code>comment_chars</code>	Set the "comment" character(s) in MTA configuration files
<code>content_return_block_limit</code>	Force NOTARY non-return of content flag for messages over the specified size
<code>conversion_size</code>	Set the number of entries allowed in the conversion file (legacy configuration) or conversions MTA option (Unified Configuration)
<code>defer_group_processing</code>	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
<code>delimiter_char</code>	(New in 7.0) RESTRICTED. Specify the delimiter character (normally the vertical bar character)
<code>delivery_options</code>	Specify interpretation of mailDeliveryOption values
<code>dequeue_debug</code>	Enable debugging of message dequeue operations
<code>dequeue_map</code>	RESTRICTED. Map files into memory when dequeuing
<code>describe_cache_limit</code>	Control size of message part description cache for message body processing purposes
<code>digest_on</code>	Specify the comment string that enables mailing list digests (in preference to regular mailing list postings); usage is RESTRICTED
<code>dirsync_hack</code>	In dirsync mode, enable "hack" of falling back to trying to do an LDAP lookup and adding the LDAP result to the alias database; usage is DELETED
<code>domain_failure</code>	Rewrite template to apply when an LDAP lookup encounters an LDAP error
<code>domain_hash_size</code>	Set the number of rewrite rules allowed
<code>domain_match_cache_size</code>	Number of domain lookup (\$V rewrite template) results to keep cached
<code>domain_match_cache_timeout</code>	How long to cache domain lookup (\$V rewrite template) results
<code>domain_match_url</code>	URL for an extra, special lookup for domains, <i>e.g.</i> , URL for looking up vanity domains
<code>domain_uplevel</code>	Control whether to search "upwards" in the DC tree for domain names, and whether to use a "canonical" domain name when searching for user entries

MTA options listed alphabetically

<code>duplicate_tracking_url</code>	(New in 8.0) Template for location of per-user duplicate message tracking files
<code>enable</code>	Enable operation of the MTA
<code>enable_sieve_regex</code>	Control whether Sieve filters may use the regex extension (the <code>:regex</code> match-type)
<code>error_text_*</code>	Specify alternate error text for any of various error conditions; see Table of error text options
<code>expandable_default</code>	Set the default for mailing lists to [NONEXPANDABLE], <i>i.e.</i> , disable use of SMTP EXPN
<code>exproute_forward</code>	Control whether the exproute channel option affects forward pointing headers
<code>file_member_size</code>	Maximum number of configuration files used by the MTA
<code>filter_cache_size</code>	New in 6.2-0.01 . Specify the size of the per-process cache of tokenized Sieve filters
<code>filter_cache_timeout</code>	New in 6.2-0.01 . Specify the retention time for entries in the per-process cache of tokenized Sieve filters
<code>filter_debug</code>	New in 6.2-0.01 . Control whether detailed (stack state) debugging of Sieve evaluation is output.
<code>filter_discard</code>	Control whether messages discarded by a Sieve filter are immediately deleted, or instead routed to the filter_discard channel for delayed deletion
<code>filter_jettison</code>	New in 6.1-0.01 . Control whether messages jettisoned by a Sieve filter are immediately deleted, or instead routed to the filter_discard channel for delayed deletion
<code>forward_data_size</code>	Hash size for the forward (database) text file
<code>fruits_size</code>	Number of fruits allowed in the fruit validation table
<code>fsync</code>	Do an fsync upon file close
<code>general_data_size</code>	Hash size for the general (database) text file
<code>group_dn_template</code>	Template used when looking up a uniqueMember of a group
<code>header_limit</code>	Sets a maximum size for the primary (outermost) message header
<code>held_sndopr</code>	Send operator or syslog messages when messages are HELD
<code>history_to_return</code>	Control the amount of delivery attempt history included in bounced messages
<code>host_hash_size</code>	Set the number of channel host names
<code>id_domain</code>	Set the domain name used in constructing message IDs

idn_config_file	(New in 8.0.1) Specify the location of optional IDNKIT configuration file
improute_forward	Control the effect of the improute channel option on forward pointing headers
include_connectioninfo	New in 6.2-0.01 . Include transport and application connection information in mapping table probes
include_conversiontag	New in JES MS 6.3 . Include the conversion tag value(s) in mapping table probes
ldap_*	Specify an alternate attribute name for any of various per-user attributes; see Direct LDAP attribute name MTA options
ldap_attr_domain1_schema2	Specify an alternate attribute name for the attribute used to store the "primary" domain in Schema 2
ldap_attr_domain2_schema2	Specify an alternate attribute name for the attribute used to store the "secondary" domain in Schema 2
ldap_attr_domain_search_filter	Attribute in the configuration template area, (see the ldap_global_config_templates MTA option) that is used to store the domain search filter template
ldap_attr_name_hash_size	Internal hash size for the list of LDAP attributes relevant to the MTA
ldap_default_attr	For LDAP URLs that are supposed to return a single result, specify a single attribute to request if no attribute was specified in the original LDAP query string
ldap_default_domain	The default domain name; overrides the <code>service.defaultdomain</code> configutil parameter (legacy configuration) or defaultdomain base option (Unified Configuration)
ldap_domain_attr_*	Specify an alternate attribute name for any of various per-domain attributes; see MTA LDAP attribute name options
ldap_domain_filter_schema1	When schema 1 is in use, specify the filter used to find domains in the DIT
ldap_domain_filter_schema2	When schema 2 is in use, specify the filter used to find domains in the DIT
ldap_domain_known_attributes	Control whether the MTA fetches all domain attributes <i>vs.</i> fetches only "known" domain attributes
ldap_domain_root	The base DN for the domain portion of the DIT; overrides the <code>service.droot</code> configutil parameter (legacy configuration) or droot base option (Unified Configuration)
ldap_domain_timeout	New in 6.1-0.01 . Specify the retention time for entries in the domain map cache

MTA options listed alphabetically

<code>ldap_global_config_templates</code>	Schema 2 support: specify the DN where global configuration templates can be found
<code>ldap_group_object_classes</code>	The object classes required for a group
<code>ldap_host</code>	Host to which to connect for LDAP queries; overrides the <code>local.ugldaphost</code> configutil parameter (legacy configuration) or <code>ugldaphost</code> base option (Unified Configuration)
<code>ldap_host_alias_list</code>	Local host aliases; overrides (for MTA purposes) the <code>local.imta.hostnamealiases</code> configutil parameter (legacy configuration) or <code>ldap_host_alias_list</code> base option (Unified Configuration)
<code>ldap_local_host</code>	The local host name (official host name for the "I" channel); overrides (for MTA purposes) the <code>local.hostname</code> configutil parameter (legacy configuration) or <code>hostname</code> base option (Unified Configuration)
<code>ldap_mail_aliases</code>	The attributes in which aliases are stored; overrides the <code>local.imta.mailaliases</code> configutil parameter
<code>ldap_mail_reverses</code>	The attributes used for the filter generated by the <code>\$Q LDAP substitution sequence</code> ; normally used during <code>reverse_url</code> lookups, hence normally the attributes compared against an address to be "reversed"
<code>ldap_max_connections</code>	The maximum number of simultaneous LDAP connections to allow
<code>ldap_object_class_hash_size</code>	Internal hash size for the list of object classes relevant to the MTA
<code>ldap_pab_host</code>	New in 7.0. Usage: RESTRICTED. The host for PAB LDAP queries; overrides (for MTA purposes) the <code>local.service.pab.ldaphost</code> configutil parameter (legacy configuration) or <code>ldaphost PAB option</code> (Unified Configuration)
<code>ldap_pab_password</code>	New in 7.0. Usage: RESTRICTED. The password for PAB LDAP queries; overrides (for MTA purposes) the <code>local.service.pab.ldappasswd</code> configutil parameter (legacy configuration) or <code>ldappasswd PAB option</code> (Unified Configuration)
<code>ldap_pab_port</code>	New in 7.0. Usage: RESTRICTED. The port for PAB LDAP queries; overrides (for MTA purposes) the <code>local.service.pab.ldappport</code> configutil parameter (legacy configuration) or <code>ldappport PAB option</code> (Unified Configuration)
<code>ldap_pab_username</code>	New in 7.0. Usage: RESTRICTED. The username (bind credentials) for PAB LDAP queries; overrides (for MTA purposes) the <code>local.service.pab.ldaphbinddn</code> configutil

	parameter (legacy configuration) or ldapbinddn PAB option (Unified Configuration)
ldap_password	The password to use when binding for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapbindcred</code> configutil parameter (legacy configuration) or ugldapbindcred base option (Unified Configuration)
ldap_port	Port to which to connect for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapport</code> configutil parameter (legacy configuration) or ugldapport base option (Unified Configuration)
ldap_schemalevel	The schema level in use; overrides (for MTA purposes) the ldap_schemalevel base option
ldap_schematag	The tag (name) of the schema in use; overrides the <code>local.imta.schematag</code> configutil parameter
ldap_timeout	Timeout value for LDAP queries
ldap_uid_invalid_chars	Characters that are not allowed in a uid
ldap_use_async	Control use of asynchronous (<i>vs.</i> synchronous) LDAP lookups
ldap_user_object_classes	The object classes required for a user
ldap_user_root	The base DN for the user and group portion of the DIT; overrides (for MTA purposes) the <code>local.ugldapbasedn</code> configutil parameter (legacy configuration) or ugldapbasedn base option (Unified Configuration)
ldap_username	The DN under which to bind for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapbinddn</code> configutil parameter (legacy configuration) or ugldapbinddn base option (Unified Configuration)
line_limit	Limit the size of messages allowed through the MTA
lines_to_return	Lines included when returning samples of message content (as in warning messages)
log_alq	(OpenVMS only.) Specify the default allocation quantity for the MTA message transaction log file
log_auth	(New in 7.0.5.) Include SMTP MAIL FROM's AUTH parameter in MTA message transaction log entries.
log_callout_delays	(New in 8.0) Include timers showing the time for responses from external components in MTA message log entries
log_connection	Generate connection log entries, and/or include connection information in message transaction log entries. (New in 6.2-0.01, optionally log SMTP AUTH attempts.)

MTA options listed alphabetically

<code>log_connections_syslog</code>	Send connection log entries to syslog (UNIX) or event log (NT)
<code>log_conversion_tag</code>	(New in 7.0.5.) Include <code>conversion tag(s)</code> field in MTA message transaction log entries.
<code>log_delay_bins</code>	Specify the bins for delivery delay range counters
<code>log_delivery_flags</code>	(New in 7.0.5.) Include delivery flag in MTA message transaction log entries.
<code>log_envelope_id</code>	New in 6.1-0.01. Include envelope ids in MTA message log entries.
<code>log_filename</code>	Include message file names in MTA message log entries
<code>log_filter</code>	Include filter actions applying to the message in MTA message log entries. (Enhanced effect in 6.2-0.01.)
<code>log_format</code>	Control the format of the MTA message log file
<code>log_header</code>	Include message headers in MTA message log entries
<code>log_imap_flags</code>	(New in 7.0.5.) Include any <code>IMAP flags set by the MTA</code> in MTA message transaction log entries
<code>log_intermediate</code>	New in 6.2-0.01. Include "intermediate" and/or "original" (RCPT TO: command line) forms of destination address in MTA message log entries
<code>log_local</code>	Include the local domain name on "bare user name" addresses in MTA message log entries
<code>log_mailbox_uid</code>	Include the IMAP UID and UIDVALIDITY of messages delivered by the <code>ims-ms</code> channel to the Message Store in MTA message log entries
<code>log_message_id</code>	Include message IDs in MTA message log entries
<code>log_messages_syslog</code>	Send MTA message log file entries to syslog (UNIX) or event log (NT)
<code>log_node</code>	OpenVMS only. Include the node name for an enqueueing process in MTA message log entries
<code>log_notary</code>	Include a NOTARY (delivery receipt) flags indicator in MTA message log entries
<code>log_priority</code>	Include message priority in MTA message log entries
<code>log_process</code>	Include enqueueing process ID in MTA message log entries
<code>log_reason</code>	Include the reason for a message rejection in MTA message log entries
<code>log_sensitivity</code>	Include message's sensitivity value in MTA message log entries
<code>log_size_bins</code>	Specify the bins for message size range counters

<code>log_sndopr</code>	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
<code>log_times</code>	(New in 8.0) Include deferred delivery time in MTA message log entries
<code>log_tracking</code>	(New in 8.0) Include message tracking ID in MTA message log entries
<code>log_transaction_delays</code>	(New in 8.0.1) Not yet fully implemented.
<code>log_transactionlog</code>	(New in 8.0) Include string argument from any Sieve "transactionlog" actions in MTA message log entries
<code>log_uid</code>	(New in 8.0) Include recipient UID in MTA message log entries
<code>log_use_xtext</code>	(New in MS 8.0) Controls xtext encoding of addresses in MTA message transaction log entries
<code>log_username</code>	Include user identity information in MTA message log entries
<code>mail_off</code>	Specify the comment string that disables mail delivery for list addresses
<code>map_names_size</code>	Set the number of mapping tables
<code>mapping_paranoia</code>	(New in 7.0) Control handling of vertical bar characters in FROM_ACCESS mapping table and recipient access mapping table and auth_rewrite mapping table probes
<code>max_addheaders</code>	Maximum number of Sieve "addheader" actions that can be performed
<code>max_alias_levels</code>	Set the level of alias nesting allowed
<code>max_duplicates</code>	(New in 8.0) Maximum "duplicate" tests performed in a Sieve filter
<code>max_fileintos</code>	Maximum number of Sieve "fileinto" actions that may be performed by a Sieve script
<code>max_header_block_user</code>	Fine tune message fragmentation
<code>max_header_line_user</code>	Fine tune message fragmentation
<code>max_internal_blocks</code>	Specify size of messages beyond which to buffer to temporary files
<code>max_local_received_lines</code>	Occurrences of the local host name in Received: header lines after which a message will be HELD
<code>max_mime_levels</code>	Degree to look inside MIME messages during processing
<code>max_mime_parts</code>	Number of parts to look at when processing MIME messages
<code>max_mr_received_lines</code>	Number of MR-Received: header lines after which a message will be HELD
<code>max_received_lines</code>	Number of Received: header lines after which a message will be HELD

MTA options listed alphabetically

<code>max_redirects</code>	Maximum number of Sieve "redirect" actions (<i>i.e.</i> , forwards) that may be used in a mailbox filter
<code>max_total_received_lines</code>	Number of Received:, MR-Received: or X400-Received: header lines after which a message will be HELD
<code>max_urls</code>	Maximum number of URLs that may be active (nesting of references)
<code>max_vacations</code>	Maximum number of vacation actions that may appear in a Sieve filter
<code>max_variables</code>	Maximum number of variables that may be used in a Sieve script
<code>max_x400_received_lines</code>	Number of X400-Received: header lines after which a message will be HELD
<code>memcache_expire</code>	(New in 8.0) Time to hold idle memcached connections open
<code>memcache_host</code>	(New in 8.0) Host name of memcached server
<code>memcache_port</code>	(New in 8.0) memcached service port
<code>message_hash_algorithm</code>	(New in JES MS 6.3-0.15) Algorithm to use for generating message hashes
<code>message_hash_fields</code>	(New in JES MS 6.3) Header fields to include when generating message hashes
<code>message_save_copy_flags</code>	(New in JES MS 6.3) Control the format of MESSAGE-SAVE-COPY mapping table probes
<code>missing_recipient_group_text</code>	Phrase to use when generating an empty group construct
<code>missing_recipient_policy</code>	Legalize messages that lack any recipient headers
<code>name_table_name</code>	OpenVMS only: Name of a logical name table storing aliases
<code>normal_block_limit</code>	Maximum size of message to treat as being of normal or higher priority
<code>non_urgent_block_limit</code>	Maximum size of message to treat as being of non-urgent priority
<code>notary_decode</code>	Control whether encoded-words in the original message header are decoded when performing a %H substitution during the MTA's generation of DSNs
<code>notary_quote</code>	Specify the character that marks substitution sequences in return_*.txt files and disposition_*.txt files
<code>notify_maximum_timeout</code>	(New in MS 7.0.5) Specify the maximum timeout permitted between successive Sieve notify or enotify actions
<code>notify_minimum_timeout</code>	(New in MS 7.0.5) Specify the minimum timeout permitted between successive Sieve notify or enotify actions

notify_timeout_default	(New in MS 7.0.5) Specify the default timeout permitted between successive Sieve notify or enotify actions
optin_user_carryover	New in 6.2-0.01. Specify whether user spam/virus filter "optin" is considered to "carry over" for forwarded recipients
options_hash_size	Hash size for the internal table of MTA options (<code>option.dat</code> options)
or_clauses	Set default for whether to AND or OR multiple posting access control settings on mailing lists
post_off	Specify the comment string that disables posting for list addresses
plaintextmincipher	(New in MS 7.4-18.01; only settable at the MTA level in Unified Configuration) Disable PLAINTEXT SMTP AUTH unless SSL or TLS is active
projectid	(New in MS 7.3-11.01) Numeric id the MTA uses when obtaining shared memory segments
queue_cache_mode	Tells the MTA where queue cache information is being stored
received_domain	Specify the domain name (identifying the system itself) to use in constructing Received: header lines
received_version	Specify the IMTA version string to use in constructing Received: header lines; use is NOT RECOMMENDED
return_address	Set the return address for the local postmaster
return_debug	Enable debugging of MTA periodic return job operations
return_delivery_history	Control whether delivery attempt history is included in returned messages
return_envelope	Control use of empty return address in notification messages, and validity checks on envelope From address
return_personal	Set the personal name for the postmaster
return_units	Control the assumed units for the notices channel option, thereby controlling the interval at which certain notification messages are generated
reverse_address_cache_size	Number of LDAP address reversal (reverse_url) results to keep cached
reverse_address_cache_timeout	How long to cache LDAP address reversal lookup (reverse_url) results
reverse_data_size	Internal hash size for the reverse (database) text file
reverse_envelope	RESTRICTED: Control application of address reversal to envelope addresses
reverse_url	URL for doing address reversal

MTA options listed alphabetically

route_to_routing_host	Forcibly route non-local hosts to the first value of mailRoutingHosts
separate_connection_log	Write connection transaction log entries to a separate file than message transaction log entries
sieve_redirect_add_resent	(New in JES MS 6.3p1) Default for whether Sieve filter "redirect" actions cause addition of Resent-* header lines
sieve_user_carryover	Specify whether user Sieve filters "carry over" for forwarded recipients
smtpproxypassword	(New in MS 7.0.5) Password the MMP uses to authenticate for SMTP; (supercedes the older TCP/IP-channel-specific option PROXY_PASSWORD)
sndopr_priority	Set the priority of operator broadcast or the syslog level of syslog messages
spamfilterN_action_M	URL that resolves to a Sieve filter, specifying the action to take in when Spam/virus filter package returns the corresponding spamfilterN_verdict_M verdict
spamfilterN_config_file	Location of the Spam/virus filter package configuration file
spamfilterN_final	Control whether the MTA passes the "intermediate" <i>vs.</i> "final" address form to the spam/virus filter package
spamfilterN_library	Location of the Spam/virus filter package client shared library
spamfilterN_null_action	URL that resolves to a Sieve filter, specifying the action to take in the case of a null verdict from Spam/virus filter package
spamfilterN_null_optin	New in 6.1-0.01. Specify a value for the attribute named by the ldap_domain_attr_optin option or ldap_optin option that means that the attribute should be ignored; the purpose is to allow a way of "turning off" spam filtering for sites and users whose provisioning facilities find it easier to set an attribute value rather than to remove an attribute entirely
spamfilter_string_action	URL that resolves to a Sieve filter, specifying the action to take in the case of verdicts from Spam/virus filter package that do not have an explicit corresponding action
spamfilterN_verdict_M	A Spam/virus filter package verdict string
sslnicknames	(New in MS 7.4-18.01; only settable at the MTA level in Unified Configuration) SSL/TLS server certificate nicknames the MTA will offer
string_pool_size_0	Set the number of strings allowed for miscellaneous MTA configuration use

<code>string_pool_size_1</code>	Set the number of strings allowed for MTA mapping table use
<code>string_pool_size_2</code>	Set the number of strings allowed for MTA alias use
<code>string_pool_size_3</code>	Set the number of strings allowed for MTA general (database) text file use
<code>string_pool_size_4</code>	Set the number of strings allowed for MTA personal conversion use
<code>subaddress_char</code>	Specify the character used in addresses to separate the username (mailbox) from the subaddress or foldername
<code>token_char</code>	Specify token character
<code>tracking_debug</code>	(New in 8.0) Enable debugging of message tracking
<code>tracking_mode</code>	(New in 8.0) Enable message tracking data storage
<code>tracking_retries</code>	(New in 8.0) Number of tracking update retry attempts
<code>tracking_retry_delay</code>	(New in 8.0) Time delay between tracking update retry attempts
<code>unique_id_template</code>	(New in JES MS 6.3) Specify a template used to convert an address into a unique identifier; typically used in conjunction with archiving facilities
<code>urgent_block_limit</code>	Maximum size of message to treat as being of urgent priority
<code>use_alias_database</code>	Control use of the alias database
<code>use_comment_strings</code>	Control the format of comment_strings mapping table probes
<code>use_domain_database</code>	Control use of the domain database
<code>use_forward_database</code>	Control use of the forward database
<code>use_personal_aliases</code>	Control use of personal alias databases
<code>use_personal_names</code>	Control the format of personal_names mapping table probes
<code>use_permanent_error</code>	Control whether certain error conditions are considered to be permanent errors, or temporary errors
<code>use_reverse_database</code>	Control use and format of the reverse mapping table and reverse database
<code>use_temporary_error</code>	(New in 8.0.1) Control whether certain error conditions are considered to be temporary errors, or permanent errors
<code>use_text_databases</code>	Control whether the general database , the reverse database, and the forward database are on-disk databases, or in-memory structures
<code>url_result_cache_size</code>	Number of rewrite rule and mapping table LDAP URL lookups (\$)[...[lookups) to keep cached

MTA options listed by functional group

<code>url_result_cache_timeout</code>	How long to cache rewrite rule and mapping table LDAP URL lookups
<code>user_case</code>	Lower case postmaster; other usernames with localbehavior channel option
<code>vacation_cleanup</code>	Average number of times between cleanups of the per-user per-response vacation files
<code>vacation_maximum_timeout</code>	(New in MS 7.0.5) Specify a maximum value permitted for Sieve "vacation" periodicity arguments
<code>vacation_minimum_timeout</code>	(New in MS 7.0.5) Specify a minimum value permitted for Sieve "vacation" periodicity arguments
<code>vacation_template</code>	URL specifying where per-user per-response vacation information is stored
<code>wild_pool_size</code>	Set the total number of wildcards allowed in mappings' patterns

39.5 MTA options listed by functional group

[Table of MTA options, listed by functional group](#), shown below, lists the available MTA options, grouped by functionality. A brief description of each option is included, though more details can be found in the discussions of each individual MTA option, and in the discussions of functional groups of MTA options. See also [Table of MTA options, listed alphabetically](#), which lists the MTA options alphabetically.

Options that fall into more than one category (as is frequently the case) are listed under each of the groups, *except* for the LDAP attribute name (schema) options which, due to their extensive number, are listed only in the LDAP attribute name (schema) options section of the table.

Table 39.2 MTA options, listed by functionality

Option	Usage
<code>enable</code>	Enable operation of the MTA
Alias and address MTA options	
<code>alias_case</code>	RESTRICTED: Control the case sensitivity of aliases
<code>alias_domains</code>	Control the format of alias file , alias option , and alias database probes
<code>alias_magic</code>	RESTRICTED: Control the order in which alias lookups are performed
<code>ap_debug</code>	Internal MTA debugging option: enable debugging of low level address parsing; support for this option is RESTRICTED
<code>delimiter_char</code>	(New in MS 7.0) RESTRICTED: ASCII position of delimiter character
<code>exproute_forward</code>	Control whether the exproute channel option affects forward pointing headers

improute_forward	Control the effect of the improute channel option on forward pointing headers
local_format_restrictions	Restrict use of the vertical bar (pipe) character, , in local mailboxes (usernames), and restrict delivery to local files (use of the +filename syntax)
max_alias_levels	Set the level of alias nesting allowed
missing_recipient_group_text	Phrase to use when generating an empty group construct
missing_recipient_policy	Legalize messages that lack any recipient headers
name_table_name	OpenVMS only: Name of a logical name table storing aliases
reverse_envelope	Control application of address reversal to envelope addresses
subaddress_char	Specify the character used in addresses to separate the username (mailbox) from the subaddress or foldername
token_char	Specify token character in local-part of address for SRS purposes
use_alias_database	Control use of the alias database
use_domain_database	Control use of the domain database
use_forward_database	Control use of the forward database
use_personal_aliases	Control use of personal alias databases
use_reverse_database	Control use and format of the REVERSE mapping table and reverse database
user_case	Lower case postmaster; other usernames with localbehavior channel option
Autoresponse periodicity MTA options	
autoreply_timeout_default	Specify a default duration between successive vacation responses to the same sender
notify_maximum_timeout	(New in MS 7.0.5) Specify the maximum timeout permitted between successive Sieve notify or enotify actions
notify_minimum_timeout	(New in MS 7.0.5) Specify the minimum timeout permitted between successive Sieve notify or enotify actions
notify_timeout_default	(New in MS 7.0.5) Specify the default timeout permitted between successive Sieve notify or enotify actions
vacation_cleanup	Average number of times between cleanups of the per-user per-response vacation files
vacation_maximum_timeout	(New in MS 7.0.5) Specify a maximum value permitted for Sieve "vacation" periodicity arguments

MTA options listed by functional group

vacation_minimum_timeout	(New in MS 7.0.5) Specify a minimum value permitted for Sieve "vacation" periodicity arguments
vacation_template	URL specifying where per-user per-response vacation information is stored
BURL MTA options	
imap_password	(New in 7.0.) Set the password for the MTA to use when connecting to the IMAP server (for BURL purposes)
imap_username	(New in 7.0.) Set the username for the MTA to use when connecting to the IMAP server (for BURL purposes)
configutil override MTA options +	
ldap_base_dn	DELETED for MS 7.0u4; see instead ldap_user_root
ldap_default_domain	The domain name to recognize and interpret as the default domain name; overrides (for MTA purposes) the defaultdomain base option (legacy configuration <code>service.defaultdomain</code> configutil parameter)
ldap_domain_root	The base DN for the domain portion of the DIT; overrides the dcroot base option (legacy configuration) <code>service.dcroot</code> configutil parameter)
ldap_host	Host to which to connect for LDAP queries; overrides the ugldaphost base option (legacy configuration <code>local.ugldaphost</code> configutil parameter)
ldap_host_alias_list	Local host aliases; overrides (for MTA purposes) the ldap_host_alias_list base option (legacy configuration <code>local.imta.hostnamealiases</code> configutil parameter)
ldap_local_host	The local host name (official host name for the "I" channel); overrides (for MTA purposes) the hostname base option (legacy configuration <code>local.hostname</code> configutil parameter)
ldap_mail_aliases	The attributes in which aliases are stored; overrides the legacy configuration <code>local.imta.mailaliases</code> configutil parameter
ldap_pab_host	(New in MS 7.0) The host for PAB LDAP queries; overrides (for MTA purposes) the ldaphost PAB option (legacy configuration <code>local.service.pab.ldaphost</code> configutil parameter)
ldap_pab_max_connections	(New in MS 7.0u1) Limit on the maximum number of connections that will be made
ldap_pab_password	(New in MS 7.0) The password for PAB LDAP queries; overrides (for MTA purposes) the

	ldappasswd PAB option (legacy configuration <code>local.service.pab.ldappasswd</code> configutil parameter)
ldap_pab_port	(New in MS 7.0) The port for PAB LDAP queries; overrides the ldapport PAB option (legacy configuration <code>local.service.pab.ldapport</code> configutil parameter). If neither this MTA option nor the PAB option (configutil parameter) is set, then this defaults to the ldap_port value.
ldap_pab_username	(New in MS 7.0) The username (bind credentials) for PAB LDAP queries; overrides (for MTA purposes) the ldapbinddn PAB option (legacy configuration <code>local.service.pab.ldapbinddn</code> configutil parameter)
ldap_password	The password to use when binding for LDAP queries; overrides (for MTA purposes) the ugldapbindcred base option (legacy configuration <code>local.ugldapbindcred</code> configutil parameter)
ldap_port	Port to which to connect for LDAP queries; overrides (for MTA purposes) the ugldapport base option (legacy configuration <code>local.ugldapport</code> configutil parameter)
ldap_schematag	The tag (name) of the schema in use; overrides (for MTA purposes) the legacy configuration <code>local.imta.schematag</code> configutil parameter
ldap_username	The DN under which to bind for LDAP queries; overrides (for MTA purposes) the ugldapbinddn base option (legacy configuration <code>local.ugldapbinddn</code> configutil parameter)
ldap_user_root	The base DN for the user and group portion of the DIT; overrides (for MTA purposes) the ugldapbasedn base option (legacy configuration <code>local.ugldapbasedn</code> configutil parameter)
Conversions MTA options	
conversion_size	Set the number of entries allowed in the conversion file (legacy configuration) or conversions MTA option (Unified Configuration)
include_conversiontag	(New in JS MS 6.3) Include the conversion tag value(s) in mapping table probes
log_conversion_tag	(New in MS 7.0.5) Include conversion tag(s) field in MTA message transaction log entries.
original_channel_probe	RESTRICTED: Determine the "input" channel for CONVERSIONS mapping table probes performed by the conversion channel and certain other, special, channels
personal_conversion_size	RESTRICTED: Maximum personal conversion entries

MTA options listed by functional group

string_pool_size_0	Set the number of miscellaneous strings (in particular including strings in conversions entries) allowed in the MTA configuration
string_pool_size_4	Set the number of strings allowed for MTA personal conversion use
Counters MTA options	
circuitcheck_completed_bins	Specify the bins for the message circuit check message counters
enable_delay_timers	(New in MS 8.0) Enable timers measuring the total time the MTA spends waiting for responses from external components
log_delay_bins	Specify the bins for delivery delay range counters
log_frustration_limit	How many times a process will tolerate failure to be able to update the MTA counters before giving up on updating counters
log_size_bins	Specify the bins for message size range counters
log_sndopr	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
log_statistics	RESTRICTED: Normal <i>vs.</i> "string" counters creation.
Database MTA options	
alias_case	RESTRICTED: Control the case sensitivity of aliases
alias_database_url	(New in MS 8.0) memcache : URL for storing alias database data
alias_domains	Control the format of alias file , alias option , and alias database probes
alias_magic	RESTRICTED: Control the order in which alias lookups are performed
domain_database_url	(New in MS 8.0) memcache : URL for storing domain database data
forward_data_size	Hash size for the forward (database) text file
forward_database_url	(New in MS 8.0) memcache : URL for storing forward database data
general_data_size	Hash size for the general (database) text file
general_database_url	(New in MS 8.0) memcache : URL for storing general database data
queue_cache_mode	Tells the MTA where queue cache information is being stored
queue_cache_mode_3_files	(New in MS 7.4) RESTRICTED
reverse_data_size	Internal hash size for the reverse (database) text file
reverse_database_url	(New in MS 8.0) memcache : URL for storing reverse database data
ssr_database_url	(New in MS 8.0) memcache : URL for storing Server Side Rules data

use_alias_database	Control use of the alias database
use_domain_database	Control use of the domain database
use_forward_database	Control use of the forward database
use_personal_aliases	Control use of personal alias databases
use_reverse_database	Control use and format of the REVERSE mapping table and reverse database
use_text_databases	Control whether the general database , the reverse database , and the forward database are on-disk databases, or in-memory structures
vacation_template	URL specifying where per-user per-response vacation information is stored
Debug MTA options	
ap_debug	RESTRICTED: Internal MTA debugging option; enable debugging of low level address parsing
cache_debug	Output direct LDAP lookup caching statistics
config_debug	RESTRICTED: (New in MS 8.0.1) Enable debugging of MTA configuration loading
debug_flush	(New in MS 7.1) Flush certain debug output to disk.
dequeue_debug	Enable debugging of message dequeue operations
filter_debug	(New in MS 6.2) Control whether detailed (stack state) debugging of Sieve filter evaluation is output
log_debug	RESTRICTED: Enable debugging of MTA logging operations
mm_debug	Internal MTA debugging option; enable debugging of message enqueue (MM) operation
os_debug	RESTRICTED: Internal MTA debugging option; enable debugging of low level OS interface routines, <i>e.g.</i> , file operations
return_debug	Enable debugging of MTA periodic return job operations
return_verify	(New in MS 7.0.5, replacing former <code>imta_return_verify</code> MTA Tailor option) Enable shell script logging within the MTA return job
tracking_debug	RESTRICTED: (New in MS 8.0) Enable debugging of the tracking subsystem
LDAP bind and connect MTA options	
ldap_base_dn	DELETED for MS 7.0u4; see instead the ldap_user_root MTA option
ldap_host	Host to which to connect for LDAP queries; overrides (for MTA purposes) the <code>local.ugldaphost</code> <code>configutil</code> parameter (legacy configuration) or ugldaphost base option (Unified Configuration)
ldap_max_connections	The maximum number of simultaneous LDAP connections to allow

MTA options listed by functional group

ldap_password	The password to use when binding for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapbindcred</code> configutil parameter (legacy configuration) or <code>ugldapbindcred</code> base option (Unified Configuration)
ldap_port	Port to which to connect for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapport</code> configutil parameter (legacy configuration) or <code>ugldapport</code> base option (Unified Configuration)
ldap_timeout	Timeout value for LDAP queries
ldap_use_async	Control use of asynchronous (<i>vs.</i> synchronous) LDAP lookups
ldap_username	The DN under which to bind for LDAP queries; overrides (for MTA purposes) the <code>local.ugldapbinddn</code> configutil parameter (legacy configuration) or <code>ugldapbinddn</code> base option (Unified Configuration)
max_urls	Maximum number of URLs that may be active (nesting of references)
Direct LDAP domain lookup MTA options	
domain_failure	Rewrite template to apply when an LDAP domain lookup encounters an LDAP error
domain_match_cache_size	Number of domain lookup (<code>\$V rewrite template</code>) results to keep cached
domain_match_cache_timeout	How long to cache domain lookup (<code>\$V rewrite template</code>) results
domain_match_url	URL for an extra, special lookup for domains, <i>e.g.</i> , URL for looking up vanity domains
domain_uplevel	Control whether to search "upwards" in the DC tree for domain names, and whether to use a "canonical domain name" when searching for user entries
ldap_attr_domain1_schema2	Specify an alternate attribute name for the attribute used to store the "primary" domain in Schema 2
ldap_attr_domain2_schema2	Specify an alternate attribute name for the attribute used to store the "secondary" domain in Schema 2
ldap_attr_domain_search_filter	Attribute in the configuration template area, (see the <code>ldap_global_config_templates</code> MTA option) that is used to store the domain search filter template
ldap_basedn_filter_schema1	(New in JES MS 6.3) When schema 1 is in use, specify the filter used to identify domains when performing baseDN searches
ldap_basedn_filter_schema2	(New in JES MS 6.3) When schema 2 is in use, specify additional filter elements used to identify domains when performing baseDN searches
ldap_default_domain	The domain name to recognize and interpret as the default domain name; overrides (for MTA

	purposes) the <code>defaultdomain</code> base option (legacy configuration <code>service.defaultdomain</code> configutil parameter)
<code>ldap_domain_filter_schema1</code>	When schema 1 is in use, specify the filter used to find domains in the DIT
<code>ldap_domain_filter_schema2</code>	When schema 2 is in use, specify the filter used to find domains in the DIT
<code>ldap_domain_known_attributes</code>	Control whether the MTA fetches all domain attributes <i>vs.</i> fetches only "known" domain attributes
<code>ldap_domain_root</code>	The base DN for the domain portion of the DIT; overrides (for MTA purposes) the <code>dcroot</code> base option (legacy configuration) <code>service.dcroot</code> configutil parameter)
<code>ldap_domain_timeout</code>	New in MS 6.1-0.01. Specify the retention time for entries in the domain map cache
<code>ldap_host_alias_list</code>	Local host aliases; overrides (for MTA purposes) the <code>local.imta.hostnamealiases</code> configutil parameter (legacy configuration) or <code>ldap_host_alias_list</code> base option (Unified Configuration)
<code>ldap_local_host</code>	The local host name (<code>official host name</code> for the "I" channel); overrides (for MTA purposes) the <code>local.hostname</code> configutil parameter (legacy configuration) or <code>hostname</code> base option (Unified Configuration)
Direct LDAP usergroup lookup MTA options	
<code>alias_entry_cache_negative</code>	Whether to cache negative results (<i>i.e.</i> , failures) of alias lookups (<code>alias_urlN</code> lookups)
<code>alias_entry_cache_size</code>	Number of alias lookup (<code>alias_urlN</code>) results to keep cached
<code>alias_entry_cache_timeout</code>	How long to cache alias lookup (<code>alias_urlN</code>) results
<code>alias_url0</code>	URL for doing <code>alias</code> lookups
<code>alias_url1</code>	URL for doing <code>alias</code> lookups
<code>alias_url2</code>	URL for doing <code>alias</code> lookups
<code>alias_url3</code>	URL for doing <code>alias</code> lookups
<code>allow_unquoted_addrs_violate_rfc2798</code>	Look up addresses in LDAP with quotes stripped from the local part
<code>aliasdetourhost_null_optin</code>	(New in JES MS 6.2p4) Specify a value for the attribute named by the <code>ldap_detourhost_optin</code> MTA option (and new in MS 8.0, also the attribute named by the <code>ldap_domain_attr_detourhostoptin</code>) that means that the attribute should be ignored; the purpose is to allow a way of "turning off" message detouring (where the detouring is typically for purposes of spam/virus filtering) for sites and users

MTA options listed by functional group

	whose provisioning facilities find it easier to set an attribute value rather than to remove an attribute entirely
<code>defer_group_processing</code>	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
<code>group_dn_template</code>	LDAP URL template used for fetching attributes once a user specified via a uniqueMember attribute has been located
<code>ldap_default_attr</code>	For URLs that are supposed to return a single result, specify a single attribute to request if no attribute was specified in the original LDAP query string
<code>ldap_filter_reference</code>	(New in MS 6.2.) If parental controls are enabled for a user (see the ldap_parental_controls MTA option), then the attribute named by this <code>ldap_filter_reference</code> MTA option specifies the DN of the entry that contains the actual head of household filter (typically, that is, the DN of the head of household user). (The attribute within that user entry containing the filter is specified by the ldap_hoh_filter MTA option, which defaults to <code>mailSieveRuleSource</code> . The lookup requests both the filter, contained in the attribute named by the <code>ldap_hoh_filter</code> MTA option, and the owner, contained in the attribute named by the ldap_hoh_owner MTA option, which defaults to <code>mail</code> .)
<code>ldap_group_object_classes</code>	The object classes required for a group
<code>ldap_hoh_filter</code>	(New in MS 6.2.) Specify an attribute to request when performing a head of household filter lookup.
<code>ldap_hoh_owner</code>	(New in MS 6.2.) Attribute in which to find the owner of the HOH Sieve
<code>ldap_mail_aliases</code>	The attributes in which aliases are stored; overrides the legacy configuration <code>local.imta.mailaliases</code> configutil parameter
<code>ldap_mail_reverses</code>	The attributes used for the filter generated by the <code>\$Q</code> LDAP substitution sequence ; normally used during reverse_url lookups, hence normally the attributes compared against an address to be "reversed"
<code>ldap_parental_controls</code>	(New in MS 6.2) Specifies the name of a user or group LDAP attribute whose value can request "head of household" (a.k.a "parental controls") Sieve filtering be applied to this user's (or group's) messages. Any of the values <code>Yes</code> , <code>1</code> , or <code>true</code> is considered to be requesting parental controls .
<code>ldap_uid_invalid_chars</code>	Characters that are not allowed in a uid
<code>ldap_user_object_classes</code>	The object classes required for a user
<code>ldap_user_root</code>	The base DN for the user and group portion of the DIT; overrides (for MTA purposes) the

	ugldapbasedn base option (legacy configuration <code>local.ugldapbasedn.configutil</code> parameter)
max_alias_levels	Set the level of alias nesting allowed
max_urls	Maximum number of URLs that may be active (nesting of references)
reverse_address_cache_size	Number of LDAP address reversal (reverse_url) results to keep cached
reverse_address_cache_timeout	How long to cache LDAP address reversal lookup (reverse_url) results
reverse_url	URL for doing address reversal
Direct LDAP schema MTA options	
ldap_attr_domain_search_filter	Attribute in the global configuration template area (see the ldap_global_config_templates MTA option) that is used to store the domain search filter template; for instance, one attribute that might be used for such a purpose (hence to which this option might be set) would be <code>inetDomainSearchFilter</code> .
ldap_basedn_filter_schema1	(New in JES MS 6.3) When schema 1 is in use, specify the filter used to identify domains when performing baseDN searches
ldap_basedn_filter_schema2	(New in JES MS 6.3) When schema 2 is in use, specify additional filter elements used to identify domains when performing baseDN searches
ldap_domain_filter_schema1	When schema 1 is in use, specify the filter used to find domains in the DIT
ldap_domain_filter_schema2	When schema 2 is in use, specify the filter used to find domains in the DIT
ldap_domain_known_attributes	Control whether the MTA fetches all domain attributes <i>vs.</i> fetches only "known" domain attributes
ldap_domain_root	The base DN for the domain portion of the DIT; overrides (for MTA purposes) the dcroot base option (legacy configuration) <code>service.dcroot.configutil</code> parameter)
ldap_global_config_templates	Schema 2 support: specify the DN where global configuration templates can be found
group_dn_template	LDAP URL template used for fetching attributes once a user specified via a uniqueMember attribute has been located
ldap_group_object_classes	The object classes required for a group
ldap_schemalevel	The schema level in use; overrides (for MTA purposes) the ldap_schemalevel base option
ldap_schematag	The tag (name) of the schema in use; overrides the <code>local.imta.schematag.configutil</code> parameter
ldap_user_object_classes	The object classes required for a user

MTA options listed by functional group

ldap_user_root	The base DN for the user and group portion of the DIT; overrides (for MTA purposes) the ugldapbasedn base option (legacy configuration <code>local.ugldapbasedn</code> configutil parameter)
Direct LDAP attribute interpretation MTA options	
aliasdetourhost_null_optin	(New in JES MS 6.2p4) Specify a value for the attribute named by the ldap_detourhost_optin MTA option (and new in MS 8.0, also the attribute named by the ldap_domain_attr_detourhostoptin) that means that the attribute should be ignored; the purpose is to allow a way of "turning off" message detouring (where the detouring is typically for purposes of spam/virus filtering) for sites and users whose provisioning facilities find it easier to set an attribute value rather than to remove an attribute entirely
allow_unquoted_addrs_violate_rfc2798	Look up addresses in LDAP with quotes stripped from the local part
capture_format_default	(New in MS 7.0u4) Specify the default "capture" format resulting from ldap_capture
delivery_options	Specify interpretation of mailDeliveryOption values
group_dn_template	LDAP URL template used for fetching attributes once a user specified via a uniqueMember attribute has been located
ldap_default_domain	The domain name to recognize and interpret as the default domain name; overrides (for MTA purposes) the defaultdomain base option (legacy configuration <code>service.defaultdomain</code> configutil parameter)
ldap_hoh_filter	(New in MS 6.2.) Specify an attribute to request when performing a head of household filter lookup.
ldap_hoh_owner	(New in MS 6.2.) Attribute in which to find the owner of the HOH Sieve
ldap_host_alias_list	Local host aliases; overrides (for MTA purposes) the <code>local.imta.hostnamealiases</code> configutil parameter (legacy configuration) or ldap_host_alias_list base option (Unified Configuration)
ldap_local_host	The local host name (official host name for the "I" channel); overrides (for MTA purposes) the <code>local.hostname</code> configutil parameter (legacy configuration) or hostname base option (Unified Configuration)
ldap_uid_invalid_chars	Characters that are not allowed in a uid

<code>optin_user_carryover</code>	(New in MS 6.2.) Specify whether user spam/virus filter "optin" is considered to "carry over" for forwarded recipients
<code>process_substitutions</code>	(New in MS 6.3-0.01) Process substitution characters in the URL values of various LDAP attributes
<code>route_to_routing_host</code>	Forcibly route non-local hosts to the first value of mailRoutingHosts
<code>sieve_user_carryover</code>	Specify whether user Sieve filters "carry over" for forwarded recipients
<code>spare_N_separator</code>	(New in MS 7.0u2) Specify the handling of multiple values of the ldap_spare_N attribute for corresponding N.
<code>vacation_maximum_timeout</code>	(New in MS 7.0.5) Specify a maximum value permitted for Sieve "vacation" periodicity arguments
<code>vacation_minimum_timeout</code>	(New in 7.0.5) Specify a minimum value permitted for Sieve "vacation" periodicity arguments
Direct LDAP attribute name MTA options	
<code>ldap_*</code>	Specify an alternate attribute name for any of various per-user attributes; see MTA LDAP attribute name options
<code>ldap_attr_domain_*</code>	Specify an alternate attribute name for any of a few fundamental defining attributes of domains; see MTA LDAP attribute name options
<code>ldap_domain_attr_*</code>	Specify an alternate attribute name for any of various per-domain attributes; see MTA LDAP attribute name options
Direct LDAP attributes returned upon authentication MTA options	
<code>ldap_auth_attr_hold_for</code>	(New in MS 8.0) Specify the name of an attribute that stores authenticated hold time
<code>ldap_auth_attr_mail_host</code>	(New in MS 8.0) Specify an alternate attribute name for the attribute used to store the mailHost for BURL purposes
<code>ldap_auth_attr_recall_secret</code>	(New in MS 8.0) Specify the name of an attribute used to store the user's recall secret
<code>ldap_auth_attr_sender</code>	(New in MS 8.0) Specify an alternate attribute name for the attribute used to store the user's canonical address
<code>ldap_auth_attr_submit_channel</code>	(New in MS 8.0) Specify an alternate attribute name for the attribute used to store the authenticated submission source channel name override
LDAP and URL lookup caching and timeout options	
<code>alias_entry_cache_negative</code>	Whether to cache negative results (<i>i.e.</i> , failures) of alias lookups (alias_urlN lookups)

MTA options listed by functional group

<code>alias_entry_cache_size</code>	Number of alias lookup (<code>alias_urlN</code>) results to keep cached
<code>alias_entry_cache_timeout</code>	How long to cache alias lookup (<code>alias_urlN</code>) results
<code>cache_debug</code>	Output direct LDAP lookup caching statistics
<code>domain_match_cache_size</code>	Number of domain lookup (<code>\$V rewrite template</code>) results to keep cached
<code>domain_match_cache_timeout</code>	How long to cache domain lookup (<code>\$V rewrite template</code>) results
<code>filter_cache_size</code>	(New in JES MS 6.2.) Specify the size of the per-process cache of tokenized Sieve filters
<code>filter_cache_timeout</code>	(New in JES MS 6.2.) Specify the retention time for entries in the per-process cache of tokenized Sieve filters
<code>ldap_domain_timeout</code>	New in MS 6.1-0.01. Specify the retention time for entries in the domain map cache
<code>ldap_timeout</code>	Timeout value for LDAP queries
<code>ldap_use_async</code>	Control use of asynchronous (<i>vs.</i> synchronous) LDAP lookups
<code>reverse_address_cache_size</code>	Number of LDAP address reversal (<code>reverse_url</code>) results to keep cached
<code>reverse_address_cache_timeout</code>	How long to cache LDAP address reversal lookup (<code>reverse_url</code>) results
<code>url_result_cache_size</code>	Number of rewrite rule LDAP URL lookups and mapping table LDAP URL lookups (<code>\$] . . . [lookups</code>) to keep cached
<code>url_result_cache_timeout</code>	How long to cache results of rewrite rule LDAP URL lookups and mapping table LDAP URL lookups (<code>\$] . . . [lookups</code>)
Directory location MTA options	
<code>langdir</code>	(New in MS 7.0) Specify the location of the default set of language-specific files
<code>tmpdir</code>	(New in MS 7.0) Specify the location of the directory for storing temporary files
DKIM MTA options	
<code>dkim_ignore_domains</code>	(New in MS 7.0.5) Domains to ignore in DKIM-Signature: fields
<code>dkim_preserve_domains</code>	(New in MS 7.0.5) Domains in DKIM-Signature: fields that trigger passthrough mode
<code>dkim_remove_domains</code>	(New in MS 7.0.5) Domains in DKIM-Signature: fields triggering removal of the header
DNS lookup MTA options	
<code>blocked_mail_from_ips</code>	(New in JES MS 6.1) Specify IP address literals whose A records should be ignored for purposes of the

	lookups done due to the mailfromdnsverify channel option
return_envelope	Control use of empty return address in notification messages, and validity checks on envelope From address
Error text and error interpretation MTA options	
access_errors	Control the information issued in certain error messages
error_text_*	Specify alternate error text for any of various error conditions; see error_text_* MTA options
missing_recipient_group_text	Phrase to use when generating an empty group construct
spamfilterN_optional	What to do if a spam/virus filter package is not responding: temporarily reject message <i>vs.</i> allow to pass unfiltered, plus whether to send syslog notice
use_permanent_error	Control whether certain error conditions are considered to be permanent errors, or temporary errors
use_temporary_error	(New in MS 8.0) Control whether certain error conditions are considered to be temporary errors, or permanent errors
External filtering context MTA options	
scan_channel	(New in MS 7.0.5) Channel to consider as source channel when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
scan_originator	(New in MS 7.0.5) Address to consider as the MAIL FROM/envelope From address when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
scan_recipient	(New in MS 7.0.5) Address to consider as the RCPT TO/envelope To address when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
File format MTA options	
buffer_size	Set the buffer size used when writing files; default is 8192
cache_magic	Obsolete PMDF option, controlling message file sorting
cbt	Obsolete PMDF option, to control the use of "contiguous best try" filesystem storage
comment_chars	Set the "comment" character(s) in MTA configuration files
debug_flush	(New in MS 7.1) Flush certain debug output to disk.

MTA options listed by functional group

dequeue_map	RESTRICTED. Map files into memory when dequeuing
fdirectory	OpenVMS only.
file_member_size	Maximum number of configuration files used by the MTA
fsync	Do an fsync upon file close
log_alq	(OpenVMS only) Specify the default allocation quantity for the MTA message transaction log file
log_deq	(OpenVMS only) Specify the default extend quantity for the MTA message transaction log file
max_internal_blocks	Specify size of messages beyond which to buffer to temporary files
mm_mbc	(OpenVMS only) Set the RMS RAB MBC field; default is 4
mm_mbf	(OpenVMS only) Set the RMS RAB MBF field; default is 16
notary_quote	Specify the character that marks substitution sequences in return_*.txt files and disposition_*.txt files
os_debug	RESTRICTED: Internal MTA debugging option; enable debugging of low level OS interface routines, <i>e.g.</i> , file operations
osync	(New in MS 7.0.5) Specify whether or not to set the O_SYNC flag when creating message queue files on disk.
projectid	(New in MS 7.3-11.01) Numeric id MTA uses when obtaining shared memory segments
queue_cache_mode	RESTRICTED: Tells the MTA where queue cache information is being stored
queue_cache_mode_3_files	(New in MS 7.4) RESTRICTED
use_text_databases	Control whether the general database , the reverse database , and the forward database are on-disk databases, or in-memory structures
Internal size MTA options	
alias_hash_size	Set the number of aliases allowed: in the alias file (legacy configuration), or via alias option (Unified Configuration)
alias_member_size	Set the number of alias expansions allowed: in the alias file (legacy configuration), or via alias option (Unified Configuration)
channel_table_size	Set the number of channels allowed in the MTA configuration
chunk_cache_limit	Specify the size of the cache of message body chunks
circuitcheck_paths_size	Number of circuit check paths (entries) to allow in the circuit check configuration file

<code>conversion_size</code>	Set the number of entries allowed in the conversion file (legacy configuration) or <code>conversions</code> MTA option (Unified Configuration)
<code>describe_cache_limit</code>	Control size of message part description cache for message body processing purposes
<code>domain_hash_size</code>	Set the number of <code>rewrite rules</code> allowed
<code>file_member_size</code>	Maximum number of configuration files used by the MTA
<code>forward_data_size</code>	Hash size for the <code>forward (database) text file</code>
<code>fruits_size</code>	Number of fruits allowed in the fruit validation table
<code>general_data_size</code>	Hash size for the <code>general (database) text file</code>
<code>host_hash_size</code>	Set the number of channel <code>host names</code>
<code>ldap_attr_name_hash_size</code>	Internal hash size for the <code>list of LDAP attributes</code> relevant to the MTA
<code>ldap_object_class_hash_size</code>	Internal hash size for the list of object classes relevant to the MTA
<code>map_names_size</code>	Set the number of <code>mapping tables</code>
<code>options_hash_size</code>	Hash size for the internal table of MTA options (<code>option.dat</code> options)
<code>personal_conversion_size</code>	RESTRICTED: Maximum personal conversion entries
<code>reverse_data_size</code>	Internal hash size for the <code>reverse (database) text file</code>
<code>string_pool_size_0</code>	Set the number of strings allowed for miscellaneous MTA configuration use
<code>string_pool_size_1</code>	Set the number of strings allowed for MTA <code>mapping table</code> use
<code>string_pool_size_2</code>	Set the number of strings allowed for MTA <code>alias</code> use
<code>string_pool_size_3</code>	Set the number of strings allowed for MTA <code>general (database) text file</code> use
<code>string_pool_size_4</code>	Set the number of strings allowed for MTA personal conversion use
<code>wild_pool_size</code>	Set the total number of wildcards allowed in <code>mapping table patterns</code>
LDAP external directory lookup MTA options	
<code>ldap_ext_host</code>	(New in MS 7.0u2) The host for LDAP EXT queries
<code>ldap_ext_max_connections</code>	(New in MS 7.0u2) Limit on the maximum number of LDAP EXT connections
<code>ldap_ext_password</code>	(New in MS 7.0u2) The password for LDAP EXT queries
<code>ldap_ext_port</code>	(New in MS 7.0u2) The port for LDAP EXT queries
<code>ldap_ext_username</code>	(New in MS 7.0u2) The username (bind credentials) for LDAP EXT queries
LDAP PAB lookup MTA options	

MTA options listed by functional group

ldap_pab_host	(New in MS 7.0) The host for PAB LDAP queries; overrides (for MTA purposes) the ldaphost PAB option (Unified Configuration) or <code>local.service.pab.ldaphost</code> configutil parameter (legacy configuration)
ldap_pab_max_connections	(New in MS 7.0u1) Limit on the maximum number of connections that will be made
ldap_pab_password	(New in MS 7.0) The password for PAB LDAP queries; overrides (for MTA purposes) the ldappasswd PAB option (Unified Configuration) or <code>local.service.pab.ldappasswd</code> configutil parameter (legacy configuration)
ldap_pab_port	(New in MS 7.0) The port for PAB LDAP queries; overrides the <code>local.service.pab.ldapport</code> configutil parameter. If neither this option nor the configutil parameter is set, this defaults to the ldap_port value.
ldap_pab_username	(New in MS 7.0) The username (bind credentials) for PAB LDAP queries; overrides (for MTA purposes) the ldapbinddn PAB option (Unified Configuration) or <code>local.service.pab.ldapbinddn</code> configutil parameter (legacy configuration)
Mailing list and group MTA options	
defer_group_processing	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
digest_on	RESTRICTED. Specify the comment string that enables mailing list digests (in preference to regular mailing list postings)
expandable_default	Set the default for mailing lists to [NONEXPANDABLE] or in direct LDAP terms, to <code>expandable: none</code> or equivalently <code>mgmanMemberVisibility: none</code> ; <i>i.e.</i> , disable use of SMTP EXPN
mail_off	Specify the comment string that disables mail delivery for list addresses
or_clauses	Set default for whether to AND or OR multiple posting access control settings on mailing lists
post_off	Specify the comment string that disables list postings for a list address
MAILSERV MTA options	
ldap_mlsub_*	RESTRICTED: Names of LDAP attributes for MAILSERV user list subscription entries
ldap_mluser_*	RESTRICTED: Names of LDAP attributes for MAILSERV users
mailserv_*	RESTRICTED: MAILSERV admin user settings
Access mapping table MTA options	

access_auth	(New in MS 8.0) Control whether certain FROM_ACCESS mapping table probes include the SMTP MAIL FROM AUTH parameter's value
access_counts	(New in JS MS 6.3.) Control whether certain *_ACCESS mapping table probes include recipient count fields
access_errors	Control the information issued in certain error messages
access_orcpt	Control whether certain *_ACCESS mapping table probes include an ORCPT field
include_connectioninfo	(New in MS 6.2) Include transport and application connection information in various (mailing list related) mapping table probes
include_conversiontag	(New in JS MS 6.3) Include the conversion tag value(s) in mapping table probes
include_mtpriority	(New in MS 8.0) Include the value of a message's MT-PRIORITY in various mapping table probes
include_spare	(New in MS 7u2) Include the values of the LDAP attributes named by the ldap_spare_N MTA options in FROM_ACCESS and/or recipient address access mapping table probes
mapping_paranoia	(New in MS 7.0) Control handling of vertical bar characters in *_ACCESS mapping table probes
use_auth_return	(New in MS 7.0) Control use of authenticated sender address in place of the envelope From address in various address matching contexts
use_canonical_return	(New in MS 6.3) Control address reversal of envelope From address for purposes of address matching in various contexts
use_ip_access	(New in MS 7.0) Control format of probes to the IP_ACCESS mapping table
Miscellaneous mapping table MTA options	
averages_cache_size	RESTRICTED: New in MS 6.2 but not yet fully implemented. Control caching of the load average data accessed from mapping table callouts .
averages_cache_timeout	RESTRICTED: New in MS 6.2 but not yet fully implemented. Control caching of the load average data accessed from mapping table callouts .
include_conversiontag	(New in JS MS 6.3) Include the conversion tag value(s) in mapping table probes
include_mtpriority	(New in MS 8.0) Include the value of a message's MT-PRIORITY in various mapping table probes
include_spare2	(New in MS 8.0) Include the values of the LDAP attributes named by the ldap_spare_N MTA options in FORWARD mapping table probes

MTA options listed by functional group

message_save_copy_flags	(New in JES MS 6.3) Control the format of MESSAGE-SAVE-COPY mapping table probes
original_channel_probe	RESTRICTED: Determine the "input" channel for CONVERSIONS mapping table probes performed by the conversion channel and certain other, special, channels
use_comment_strings	Control the format of COMMENT_STRINGS mapping table probes
use_personal_names	Control the format of PERSONAL_NAMES mapping table probes
url_result_cache_size	Number of rewrite rule LDAP URL lookups and mapping table LDAP URL lookups (\$] . . . [lookups) to keep cached
url_result_cache_timeout	How long to cache results of rewrite rule LDAP URL lookups and mapping table LDAP URL lookups (\$] . . . [lookups)
Memcache MTA options	
alias_database_url	(New in MS 8.0) memcache: URL for storing alias database data
domain_database_url	(New in MS 8.0) memcache: URL for storing domain database data
enable_sieve_memcache	(New in MS 8.0) Control whether Sieve filters may use the private memcache extension
forward_database_url	(New in MS 8.0) memcache: URL for storing forward database data
general_database_url	(New in MS 8.0) memcache: URL for storing general database data
reverse_database_url	(New in MS 8.0) memcache: URL for storing reverse database data
ssr_database_url	(New in MS 8.0) memcache: URL for storing Server Side Rules data
memcache_expire	(New in MS 8.0) Time to hold idle memcached connections open
memcache_host	(New in MS 8.0) Host name of memcached server
memcache_port	(New in MS 8.0) Memcached service port
Message archival and hashing MTA options	
journal_format	(New in MS 7.0.5) Specify whether capture message copies generated in "journal" format are generated in 2003 <i>vs.</i> 2007 "journal" style
message_hash_algorithm	(New in MS 6.3) Algorithm to use for generating message hashes
message_hash_fields	(New in MS 6.3) Header fields to include when generating a message hash

<code>unique_id_template</code>	(New in MS 6.3) Specify a template used to convert an address into a unique identifier, typically used in conjunction with archiving facilities
Message size MTA options	
<code>block_limit</code>	Limit the size of messages allowed through the MTA
<code>block_size</code>	Set the size of MTA "blocks"
<code>bounce_block_limit</code>	Limit the amount of original message content included in bounce messages
<code>content_return_block_limit</code>	Force NOTARY non-return of content flag for messages over the specified size
<code>error_text_block_over</code>	Specify error text returned in some cases of exceeding a destination channel blocklimit
<code>error_text_line_over</code>	Specify error text returned in some cases of exceeding a destination channel linelimit
<code>error_text_list_block_over</code>	Specify error text returned in some cases of exceeding a list's [BLOCKLIMIT] or mgrpMsgMaxSize value
<code>error_text_list_line_over</code>	Specify error text returned in some cases of exceeding a list's [LINELIMIT] value
<code>error_text_user_block_over</code>	Specify error text returned in some cases of exceeding a user's mailMsgMaxBlocks value or the [BLOCKLIMIT] value for an alias
<code>error_text_user_line_over</code>	specify error text returned in some cases of exceeding a user's [LINELIMIT] value
<code>header_limit</code>	Sets a maximum size for the primary (outermost) message header
<code>line_limit</code>	Limit the size of messages allowed through the MTA
<code>max_header_block_use</code>	Fine tune message fragmentation
<code>max_header_blocks</code>	Truncate message header after the specified number of MTA blocks
<code>max_header_line_use</code>	Fine tune message fragmentation
<code>max_header_lines</code>	Truncate message header after the specified number of lines
<code>max_mime_levels</code>	Degree to look inside MIME messages during processing
<code>max_mime_parts</code>	Number of parts to look at when processing MIME messages
<code>normal_block_limit</code>	Maximum size of message to treat as being of normal or higher priority
<code>non_urgent_block_limit</code>	Maximum size of message to treat as being of non-urgent priority
<code>second_class_block_limit</code>	Maximum size of message to treat as being of second class priority
<code>urgent_block_limit</code>	Maximum size of message to treat as being of urgent priority

MTA options listed by functional group

Message tracking MTA options	
<code>log_times</code>	(New in MS 8.0) Include requested deferral time in MTA message transaction log entries
<code>log_tracking</code>	(New in MS 8.0) Include tracking ID in MTA message transaction log entries
<code>tracking_mode</code>	(New in MS 8.0) Enable/disable the MTA's tracking support
<code>tracking_debug</code>	(New in MS 8.0) Level of debug output regarding tracking
<code>tracking_retries</code>	(New in MS 8.0) How many times the MTA will reattempt a tracking update
<code>tracking_retry_delay</code>	(New in MS 8.0) The time to wait between tracking update reattempts
MeterMaid MTA options	
<code>enable_sieve_metermaid</code>	(New in MS 8.0) Control whether Sieve filters may use the private metermaid extension
<code>metermaid_backoff</code>	(New in MS 7.2)
<code>metermaid_expire</code>	(New in MS 7.2)
<code>metermaid_host</code>	(New in MS 7.2)
<code>metermaid_port</code>	(New in MS 7.2)
<code>metermaid_secret</code>	(New in MS 7.2)
<code>metermaid_timeout</code>	(New in MS 7.2)
MLS MTA options	
<code>error_text_mls_access_failure</code>	(New in MS 7.0) RESTRICTED: Not yet used
<code>ldap_mlsrange</code>	(New in MS 7.0) RESTRICTED: Not yet used
<code>mls</code>	(New in MS 7.0) RESTRICTED: Not yet fully implemented
MTQP MTA options	
<code>mtqp_expire</code>	(New in MS 8.0) Specify the MTQP expiration, in seconds
<code>mtqp_port</code>	(New in MS 8.0) Specify the MTQP port
<code>mtqp_timeout</code>	(New in MS 8.0) Specify the MTQP timeout, in seconds
Notification message MTA options	
<code>bounce_block_limit</code>	Limit the amount of original message content included in bounce messages
<code>content_return_block_limit</code>	Force NOTARY non-return of content flag for messages over the specified size
<code>history_to_return</code>	Control the amount of delivery attempt history included in bounced messages
<code>ldap_domain_attr_report_address</code>	Name of domain level LDAP attribute whose value specifies a domain-specific postmaster address

<code>lines_to_return</code>	Lines included when returning samples of message content (as in warning messages)
<code>notary_decode</code>	Control whether encoded-words in the original message header are decoded when performing a %H substitution during the MTA's generation of DSNs
<code>notary_quote</code>	Specify the character that marks substitution sequences in <code>return_*.txt</code> files and <code>disposition_*.txt</code> files
<code>return_address</code>	Set the return address for the local postmaster
<code>return_debug</code>	Enable debugging of MTA periodic return job operations
<code>return_delivery_history</code>	Control whether delivery attempt history is included in returned messages
<code>return_envelope</code>	Control use of empty return address in notification messages, and validity checks on envelope From address
<code>return_personal</code>	Set the personal name for the postmaster
<code>return_units</code>	Control the assumed units for the notices channel option, thereby controlling the interval at which certain notification messages are generated
<code>return_verify</code>	(New in MS 7.0.5, replacing former <code>imta_return_verify</code> MTA Tailor option) Enable shell script logging within the MTA return job
<code>use_precedence</code>	Control whether delayed delivery notification messages are sent for list and bulk precedence messages
<code>use_warnings_to</code>	DELETED as of MS 7.0.5: Whether to send DSNs to Warnings-to: address
Password and TLS MTA options	
<code>plaintextmincipher</code>	(New in MS 7.4-18.01; available as an MTA level option in Unified Configuration only) Disable PLAINTEXT SMTP AUTH unless SSL or TLS is active
<code>smtpproxypassword</code>	(New in MS 7.0.5) Replaces the former TCP/IP channel option file option PROXY_PASSWORD
<code>sslnicknames</code>	(New in MS 7.4-18.01; available as an MTA level option in Unified Configuration only) SSL/TLS server certificate nicknames the MTA will offer
Processing priority MTA options	
<code>defer_group_processing</code>	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
<code>log_mtpriority</code>	(New in MS 8.0) Include message MT-PRIORITY in MTA message transaction log entries
<code>log_priority</code>	Include message priority in MTA message transaction log entries

MTA options listed by functional group

<code>mtpriority_policy</code>	(New in MS 8.0) MT-PRIORITY policy name to announce in SMTP EHLO response
<code>normal_block_limit</code>	Maximum size of message to treat as being of normal or higher priority
<code>non_urgent_block_limit</code>	Maximum size of message to treat as being of non-urgent priority
<code>second_class_block_limit</code>	Maximum size of message to treat as being of second class priority
<code>urgent_block_limit</code>	Maximum size of message to treat as being of urgent priority
<code>use_precedence</code>	Control whether delayed delivery notification messages are sent for list and bulk precedence messages
Received header line MTA options	
<code>held_sndopr</code>	Send operator or syslog messages when messages are HELD
<code>id_domain</code>	Set the domain name used in constructing message IDs
<code>max_local_received_lines</code>	Occurrences of the local host name in Received: headers after which a message will be HELD
<code>max_mr_received_lines</code>	Number of MR-Received: headers after which a message will be HELD
<code>max_received_lines</code>	Number of Received: headers after which a message will be HELD
<code>max_total_received_lines</code>	Number of Received:, MR-Received: or X400-Received: headers after which a message will be HELD
<code>max_x400_received_lines</code>	Number of X400-Received: headers after which a message will be HELD
<code>received_domain</code>	Specify the domain name (identifying the system itself) to use in constructing Received: headers
<code>received_version</code>	Specify the IMTA version string to use in constructing Received: header lines; use is NOT RECOMMENDED
Sieve filter MTA options	
<code>systemfilter</code>	System Sieve filter , applied to all messages at each enqueue
Sieve filter interpretation MTA options	
<code>decode_encoded_words</code>	RESTRICTED: Decode encoded words during Sieve processing
<code>defer_header_addition?</code>	(New in MS 7.0.5.30.0) Control whether Sieve filters see any headers added prior to Sieve processing
<code>filter_discard</code>	Control whether messages discarded by a Sieve filter are immediately deleted, or instead routed to the filter_discard channel for delayed deletion

filter_jettison	New in MS 6.1. Control whether messages jettisoned by a Sieve filter are immediately deleted, or instead routed to the filter_discard_channel for delayed deletion
notify_ignore_errors	(New in MS 7.0.5) Control whether to abort or ignore invalid recipient in Sieve notify action
sieve_received	(New in MS 8.0) Make synthesized Received: header line visible for Sieve filter evaluation
sieve_redirect_add_resent	(New in JS MS 6.3p1) System default for whether Sieve "redirect" actions cause addition of Resent-* header lines
sieve_user_carryover	Specify whether user Sieve filters "carry over" for (direct LDAP mailDeliveryOption: forward) forwarded recipients
Sieve filter limit MTA options	
max_addheaders	Maximum number of Sieve "addheader" actions that can be performed
max_duplicates	(New in MS 8.0) Maximum "duplicate" tests performed in a Sieve filter
max_fileintos	Maximum number of "fileinto" actions that may be performed by a Sieve filter
max_notifys	(New in JES MS 6.2) Maximum number of Sieve "notify" actions that may be applied in a Sieve filter
max_redirect_addresses	(New in MS 7.0u1) Maximum number of addresses which will be used from a Sieve external list used in a redirect ; <i>i.e.</i> , maximum addresses that will be used from a <code>redirect :list</code> Sieve action
max_redirects	Maximum number of Sieve "redirect" actions (<i>i.e.</i> , forwards) that may be performed in a Sieve filter
max_sieve_list_size	Maximum number of elements allowed in a Sieve string-list structure
max_sieve_string_size	(New in MS 7.0u3) Maximum size allowed for any string in a Sieve script
max_vacations	Maximum number of vacation actions that may appear in a Sieve filter
max_variables	(New in JES MS 6.2) Maximum number of variables that may be used in a Sieve script
Sieve filter caching MTA options	
filter_cache_size	(New in JES MS 6.2) Specify the size of the per-process cache of tokenized Sieve filters
filter_cache_timeout	(New in JES MS 6.2) Specify the retention time for entries in the per-process cache of tokenized Sieve filters
Sieve language extension MTA options	

MTA options listed by functional group

enable_sieve_body	Control whether Sieve filters may use the body extension
enable_sieve_ereject	(New in MS 7.0u2) Control whether Sieve filters may use the ereject extension
enable_sieve_memcache	(New in MS 8.0) Control whether Sieve filters may use the private memcache extension
enable_sieve_metermaid	(New in MS 8.0) Control whether Sieve filters may use the private metermaid extension
enable_sieve_regex	Control whether Sieve filters may use the regex extension (the <code>:regex</code> match-type)
strict_require	Control whether or not to strictly enforce certain rules regarding the syntax of use of "require" in Sieve filters
Sieve filter duplicate extension MTA options	
duplicate_maximum_timeout	(New in MS 8.0) Maximum period in seconds for Sieve duplicate test storage timeout
duplicate_minimum_timeout	(New in MS 8.0) Minimum period in seconds for Sieve duplicate test storage timeout
duplicate_timeout_default	(New in) Default period in seconds for Sieve duplicate test storage timeout
duplicate_tracking_url	(New in) Template for locating per-user duplicate message information
max_duplicates	(New in MS 8.0) Maximum " duplicate " tests performed in a Sieve filter
Sieve filter error text MTA options	
error_text_sieve_access	Specify the error text used when reporting on an error accessing a user's Sieve filter file (for Sieve filters stored on disk)
error_text_sieve_syntax	Specify the error text used when reporting on a syntax error in a user Sieve filter
error_text_source_sieve_access	Specify the error text used when reporting on an error accessing a source channel Sieve filter file (for Sieve filters stored on disk)
error_text_source_sieve_syntax	(New in MS 6.3) Specify the error text used when reporting on a syntax error in a source channel Sieve filter
error_text_source_sieve_authorization	(New in MS 6.3) Specify the error text used when reporting on a general error encountered attempting to use a source channel Sieve filter
Sieve filter log and debug MTA options	
filter_debug	(New in MS 6.2) Control whether detailed (stack state) debugging of Sieve filter evaluation is output
log_filter	Include applicable Sieve filter actions in MTA message transaction log entries

log_transactionlog	(New in MS 8.0) Include Sieve "transactionlog" action strings in MTA message transaction log entries
Spamfilter MTA options	
access_errors	Control the information issued in certain error messages
brightmail_*	Aliases for corresponding spamfilter_* MTA options; see the spamfilter_* options for definitions
scan_channel	(New in MS 7.0.5) Channel to consider as source channel when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
scan_originator	(New in MS 7.0.5) Address to consider as the MAIL FROM/envelope From address when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
scan_recipient	(New in MS 7.0.5) Address to consider as the RCPT TO/envelope To address when initializing the MTA for spam/virus filter package checks performed by non-channel applications and utilities such as imexpire
error_text_spamfilterN_error	Specify default error text to return in cases of spam/virus filter package problems
ldap_domain_attr_optinN	Name of domain-level LDAP attribute selecting opt-in to spam/virus filter package N processing
ldap_optinN	Name of user-level LDAP attribute selecting opt-in to spam/virus filter package N processing
ldap_source_optinN	Name of user-level LDAP attribute selecting source opt-in to spam/virus filter package N processing
optin_user_carryover	(New in MS 6.2.) Specify whether user spam/virus filter "opt-in" is considered to "carry over" for forwarded recipients
spamfilter_*	As of JES 5/MS 6.3, obsoleted in favor of the new-in-JES-5/MS-6.3 spamfilter1_* options with which they are (became in JES 5/MS 6.3) synonymous
spamfilterN_action_M	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) URL that resolves to a Sieve filter , specifying the action to take when the Nth spam/virus filter package returns the corresponding Mth verdict, that is, the action to take corresponding to the spamfilterN_verdict_M verdict
spamfilterN_config_file	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) Location of the configuration file for the Nth spam/virus filter package

MTA options listed by functional group

spamfilterN_final	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) Control whether the MTA passes the "intermediate" <i>vs.</i> "final" address form to the Nth spam/virus filter package
spamfilterN_includeheaders	(New in MS 7.0.5) Specify whether or not to pass to the Nth (N in the range 1--8) spam/virus filter package any header lines added via the \$A flag in an address *_ACCESS mapping table
spamfilterN_library	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) Location of the client shared library for the Nth spam/virus filter package
spamfilterN_null_action	URL that resolves to a Sieve filter , specifying the action(s) to take in the case of a null verdict from the Nth spam/virus filter package
spamfilterN_null_optin	(Values for n of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) Specify a value for the attribute named by the ldap_domain_attr_optinN MTA option or ldap_optinN MTA option or (new in JES 5/MS 6.3) the ldap_source_optinN MTA option that means that the attribute should be ignored; the purpose is to allow a way of "turning off" spam filtering (by virus/spam filter package N) for sites and users whose provisioning facilities find it easier to set an attribute value rather than to remove an attribute entirely
spamfilterN_optional	What to do if a spam/virus filter package is not responding: temporarily reject message <i>vs.</i> allow to pass unfiltered, plus whether to send syslog notice
spamfilterN_received	(New in JES MS 6.2) Specify whether or not to generate a pseudo-Received: header line to pass to the spam/virus filter package.
spamfilterN_returnpath	(New in MS 7.0 update 1) Control whether or not to add a Return-path: header line to the message passed to the spam/virus filter package
spamfilterN_string_action	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) URL that resolves to a Sieve filter , specifying the action(s) to take in the case of verdicts from the Nth spam/virus filter package that do not have an explicit corresponding action
spamfilterN_verdict_M	(Values for N of 1--4 are new in JES MS 6.2; values of 5--8 are new in JES 5/MS 6.3.) For the Nth spam/virus filter package, its Mth verdict string
SPF MTA options	
error_text_spf_*	(New in MS 6.3, but not taking effect until MS 8.0) Set the text for any of several SPF-related errors
spf_max_dns_queries	(New in MS 6.3-0.15) Maximum DNS queries per SPF check

spf_max_recursion	(New in MS 6.3-0.15) Maximum recursion during SPF checks
spf_max_time	(New in MS 6.3-0.15) Maximum time (seconds) permitted for performing an SPF check
spf_smtp_status_fail	(New in MS 6.3-0.15) How to interpret an SPF domain verification failure
spf_smtp_status_fail_all	(New in MS 6.3-0.15) How to interpret an SPF subdomain verification failure
spf_smtp_status_permerror	(New in MS 6.3-0.15) Interpretation of DNS permanent errors on SPF attempts
spf_smtp_status_softfail	(New in MS 6.3-0.15) How to interpret an SPF "soft" domain verification failure
spf_smtp_status_softfail_all	(New in MS 6.3-0.15) How to interpret an SPF subdomain "soft" verification failure
spf_smtp_status_temperror	(New in MS 6.3-0.15) How to interpret DNS temporary errors during SPF lookups
SRS MTA options	
error_text_srs_*	(New in MS 7.0u2) Set the text for any of several SRS-related errors
srs_domain	(New in 6.3p1) Domain to use in SRS addresses.
srs_maxage	(New in 6.3p1) Number of days before an SRS address times out
srs_secrets	(New in 6.3p1) Secret keys used for encoding and decoding SRS addresses
token_char	Specify token character in local-part of address for SRS purposes
Syslog MTA options	
held_sndopr	Send operator or syslog messages when messages are HELD
log_connections_syslog	Send MTA connection transaction log entries to syslog (UNIX)
log_messages_syslog	Send MTA message transaction log file entries to syslog (UNIX)
log_sndopr	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
sndopr_priority	Set the priority of operator broadcast or the syslog level of syslog messages
spamfilterN_optional	What to do if a spam/virus filter package is not responding: temporarily reject message <i>vs.</i> allow to pass unfiltered, plus whether to send syslog notice
Transaction logging MTA options	
log_alq	(OpenVMS only) Specify the default allocation quantity for the MTA transaction log file(s)

MTA options listed by functional group

<code>log_auth</code>	(New in MS 7.0.5) Include SMTP MAIL FROM's AUTH parameter in MTA message transaction log entries
<code>log_callout_delays</code>	(New in MS 8.0) Include timers showing the time for responses from external components in MTA message transaction log entries
<code>log_connection</code>	Include connection information in MTA transaction log entries
<code>log_connections_syslog</code>	Send MTA connection transaction log file entries to syslog (UNIX) in addition to, or instead of to, MTA connection transaction log file
<code>log_conversion_tag</code>	(New in MS 7.0.5) Include conversion tag(s) field in MTA message transaction log entries
<code>log_debug</code>	RESTRICTED: Enable debugging of MTA logging operations
<code>log_deq</code>	(OpenVMS only) Specify the default extend quantity for the MTA transaction log file(s)
<code>log_delivery_flags</code>	(New in MS 7.0.5) Include delivery flags field in MTA message transaction log entries
<code>log_diagnostics</code>	(New in MS 7.0u1) Include diagnostics in MTA message transaction log entries
<code>log_envelope_id</code>	(New in JES MS 6.1) Include envelope id in MTA message transaction log entries
<code>log_filename</code>	Include message file names in MTA message transaction log entries
<code>log_filter</code>	Include applicable Sieve filter actions in MTA message transaction log entries
<code>log_format</code>	Control the format of the MTA transaction log file(s)
<code>log_futurerelease</code>	(New in) Include value of FUTURERELEASE SMTP extension in MTA message transaction log entries
<code>log_header</code>	Include message headers in MTA message transaction log entries
<code>log_imap_flags</code>	(New in MS 7.0.5) Include IMAP flags field in MTA message transaction log entries
<code>log_intermediate</code>	(New in MS 6.2) Include the "intermediate" address in MTA message transaction log entries
<code>log_local</code>	Include the local domain name on "bare user name" addresses in MTA message transaction log entries
<code>log_mailbox_uid</code>	(New in MS 7.0.5) Include the IMAP UID and UIDVALIDITY of messages delivered by ims-channel to the Message Store in MTA message transaction log entries
<code>log_message_id</code>	Include message IDs in MTA message transaction log entries

<code>log_messages_syslog</code>	Send MTA message transaction log file entries to syslog (UNIX) in addition to, or instead of to, MTA message transaction log file
<code>log_mtpriority</code>	(New in MS 8.0) Include message MT-PRIORITY in MTA message transaction log entries
<code>log_node</code>	(OpenVMS only prior to JES 5/MS 6.3) Include the node name on which process runs in MTA message transaction log entries
<code>log_notary</code>	Include a NOTARY (delivery receipt) flags indicator in MTA message transaction log entries
<code>log_priority</code>	Include message priority in MTA message transaction log entries
<code>log_process</code>	Include process ID in MTA message transaction log entries
<code>log_queue_time</code>	(New in JS MS 6.3) Include "time in queue" in MTA message transaction log entries ; this also causes inclusion of "time to open or fail to open a connection" in MTA connection log entries
<code>log_reason</code>	(New in JES MS 6.3) Include reason for message rejection in MTA message transaction log entries
<code>log_sensitivity</code>	Include message's sensitivity value in MTA message transaction log entries
<code>log_sndopr</code>	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
<code>log_times</code>	(New in MS 8.0) Include requested deferral time in MTA message transaction log entries
<code>log_transactionlog</code>	(New in MS 8.0) Include Sieve " transactionlog " action strings in MTA message transaction log entries
<code>log_uid</code>	(New in MS 8.0) Include recipient UIDs in MTA message transaction log entries
<code>log_use_xtext</code>	(New in MS 8.0) Controls xtext encoding of addresses in MTA message transaction log entries
<code>log_username</code>	Include the username for an enqueueing process in MTA transaction log entries
<code>return_cleanup_period</code>	Run site's cleanup script every Nth run of return_job
<code>return_split_period</code>	Start new mail.log_current file every Nth run of return_job
<code>separate_connection_log</code>	Write connection transaction log entries to a separate file than message transaction log entries
OpenVMS user agent MTA options	
<code>delivery_receipt_off</code>	(OpenVMS only) Comment string that disables delivery receipt request
<code>delivery_receipt_on</code>	(OpenVMS only) Comment string that enables delivery receipt request

<code>dis_nesting</code>	(OpenVMS only) Nesting allowed for VMS MAIL @DIS lists
<code>form_names</code>	(OpenVMS only) List of the names of pop-up form images
<code>mail_delivery_filename</code>	(OpenVMS only) Set MAIL.DELIVERY filename
<code>missing_address</code>	(OpenVMS only) Address to insert of VMS MAIL From: if empty
<code>multinet_mm_exclusive</code>	(OpenVMS only) VMS MAIL <i>vs.</i> Multinet MM mailbox
<code>read_receipt_off</code>	(OpenVMS only) Comment string that disables read receipt request
<code>read_receipt_on</code>	(OpenVMS only) Comment string that enables read receipt request
<code>safe_tcl_mode</code>	(OpenVMS only) Control handling of Safe-Tcl message parts
<code>use_mail_delivery</code>	(OpenVMS only) Enable MAIL.DELIVERY processing
<code>vms_mail_exclusive</code>	(OpenVMS only) VMS MAIL <i>vs.</i> Multinet MM mailbox

+ Note that the MTA's SMTP AUTH user authentication lookups are done using general authentication library code, also used for IMAP, POP, or mshttpd user logins (authentication). The authentication library code generally does not make use of the MTA-specific options, but rather is controlled by [Auth options](#) (or in legacy configuration, configutil parameters). However, for a few specific cases of MTA options affecting authentication library operation, see the [Direct LDAP attributes returned upon authentication MTA options](#).

39.6 enable Option Under mta

The `enable` MTA option, `mta.enable` (Unified Configuration) or `local.imta.enable` (legacy configuration), provides a default setting for the `dispatcher.enable` option and the `job_controller.enable` option. The `mta.enable` option is deprecated in favor of the two more explicit `enable` options.

The default if this option is not explicitly set is 0, but initial configuration may set this option to enable the MTA, as appropriate.

39.7 Alias and address MTA options

This discussion will focus on those MTA options that affect and modify certain fundamental aspects of MTA [alias](#) and address handling, and in particular those MTA options affecting MTA alias file or alias database handling. See also the [use_auth_return](#), [use_canonial_return](#), and [use_orig_return](#) MTA options, which among other things can affect the form of envelope From address used in [recipient-address-based *_ACCESS mapping table](#) probes and in [mailing list named parameter \[*_MAPPING\] mapping table](#) probes.

For options relating more specifically to mail group or mail list handling, (as note that groups and lists are merely special forms of alias), see also [Mailing list and group MTA options](#).

And for the (many, many) options relating more specifically to aliases stored in LDAP -- the so-called "Direct LDAP" MTA options -- see also [Direct LDAP MTA options](#).

The [ap_debug](#) MTA option enables low-level debugging (typically meaningful only to Oracle support) relating to the parsing of aliases and addresses; (the [mm_debug](#) MTA option enables somewhat higher-level debugging of address handling, which also is typically meaningful only to Oracle support.)

39.7.1 Alias and address case sensitivity option ([alias_case](#))

Use of settings other than those recommended by Oracle is RESTRICTED.

The [alias_case](#) option controls whether [aliases](#) (alias names) in the [alias database](#) or [alias file](#), or in Unified Configuration [alias options](#), are case sensitive. (It does not affect alias lookups in LDAP, that is, [alias_urlN](#) lookups: since the schema defines the `mail`, `mailAlternateAddress`, and `mailEquivalentAddress` LDAP attributes as case-insensitive, LDAP searches for these attributes are performed case-insensitively.) (Note that the MTA always preserves the case of the alias translation value, that is, the right hand side; the point of the [alias_case](#) option is to control whether the alias on the left hand side is case sensitive for matching purposes.) Note that even if aliases are case sensitive in general, [postmaster aliases](#) are always case insensitive. The default value is 0, meaning that aliases are not case sensitive. Bits 0 through 2 (values 0 through 7) control handling of alias file lookups (corresponding in Unified Configuration to alias options); higher bits control the handling of alias database lookups.

Table 39.3 [alias_case](#) MTA option values

Value	Usage
Lower bits (alias file and alias options)	
0	Case insensitive alias file aliases
1	Case sensitive alias file aliases
2	Alias file aliases are first searched for case-sensitively, but if not found, perform a case-sensitive search
3	Alias file aliases are first searched for case-sensitively, but if not found, perform a case-sensitive search
Higher bits (alias database)	
8	Case insensitive alias database aliases
9	Case sensitive alias database aliases and new in MS 8.0, case insensitive comparisons for memcache storage of the alias database (alias_database_url)
10	Alias database aliases are first searched for case-sensitively, but if not found, perform a case-sensitive search and new in MS 8.0, case insensitive comparisons for memcache storage of the alias database (alias_database_url)
11	Alias database aliases are first searched for case-sensitively, but if not found, perform a case-sensitive search and new in MS 8.0, case insensitive comparisons for memcache storage of the alias database (alias_database_url)

Bit 0 is the least significant bit.

39.7.2 Domains in alias lookups (`alias_domains`)

The `alias_domains` MTA option takes a bit encoded integer argument controlling the format of [alias file](#) (and hence in Unified Configuration [alias option](#)) and [alias database](#) lookups. (This option does not affect [alias_urlN](#) lookups.)

As of Messaging Server 7.0-3.01, the default value of `alias_domains` is 6, meaning that [alias file](#) and [alias database](#) lookups probe first with the entire address, then probe with a wildcarded localpart plus domain, and finally addresses matching the local channel probe with solely the localpart; previously, the default value had been 1, meaning that [alias file](#) and [alias database](#) lookups would probe with only the localpart (mailbox portion) of the address. Note that for addresses matching the local channel, the a localpart-only probe is made even if bit 0 (value 1) is not set. Setting bit 1 (value 2) causes a probe to be made using the entire address (including the domain name). Setting bit 2 (value 4) when bit 1 (value 2) is also set causes an additional, fall-through wildcard `*` probe to be made. Indeed, if the address included a subaddress, setting bit 2 (value 4) causes two wildcard `*` probes to be made, first `*+*@domain-name` and then `*@domain-name`. (For wildcarding solely the subaddress, not the localpart, see the [subaddresswild](#) channel option.) If all bits are set, *i.e.*, `alias_domains=7`, then the order of the probes is to first probe with the entire address (the most specific check), next probe with a wildcard `*` localpart plus the domain name, and finally probe with just the localpart.

Table 39.4 `alias_domains` MTA option bit values

Bit	Value	Usage
0	1	Look up <code>localpart</code> . Clearing this bit disables the lookup of unadorned localparts for channels other than the local channel; for the local channel, local parts are always looked up.
1	2	Look up <code>localpart@domainname</code> .
2	4	If performing entire address probes (that is, if bit 1 is also set) and if no exact match was found, try lookups with the <code>*</code> character for the username; more precisely, an <code>*+*@domainname</code> (if the address included a subaddress), and an <code>*@domainname</code> lookup.

Bit 0 is the least significant bit.

Note that by default only addresses rewritten to the local channel are checked against the [alias file](#) and [alias database](#). However, via use of the [aliaslocal channel option](#), it is possible to cause addresses matching other channels to be checked against the [alias file](#) and [alias database](#). Note that the effect of bit 2 (value 4), that is, the probes with `*` character username, can be controlled on a per-channel basis via the [aliaswild channel option](#).

39.7.3 Alias lookup control: `alias_magic` (integer)

The `alias_magic` MTA option controls the ordering of alias lookups. It takes a decimal-encoded integer argument, where each decimal digit represents a type of alias lookup, and the ordering of the digits controls the ordering of the lookups. The ones' place controls the first thing looked up; the tens' place controls the second thing looked up; the hundreds' place controls the third thing looked up; *etc.* A value of 1 means personal aliases; a value of 2 means logical aliases; a value of 3 means the [alias database](#); a value of 4 means the [alias file](#); a value of 6 means the [alias_url0 LDAP lookup](#); a value of 7 means the [alias_url1 LDAP lookup](#); a

value of 8 means the [alias_url2 LDAP lookup](#); a value of 9 means the [alias_url3 LDAP lookup](#).

Prior to 7.0U3 the default value for this option was 98764321; in 7.0U3 and later the default has been changed to the recommended operating value of 8764 for [direct LDAP](#) mode. Note that that does not enable [alias_url3 lookups](#) (value 9) or [alias database lookups](#) (value 3). An alternate, sensible value for direct LDAP *plus alias database* lookups would be

```
alias_magic=987643
```

Caution: Support for this option is **RESTRICTED**. As the `alias_magic` option affects MTA operation at a very fundamental level, in particular its fundamental means of doing alias lookups, which can have wide-ranging, both obvious and subtle effects, setting this option to other than a Oracle-engineering-recommended value is not supported.

39.7.4 Alias and address MTA options: `delimiter_char` (1-127)

RESTRICTED.

(New in 7.0.) The `delimiter_char` MTA option controls what character represents a delimiter. The value of this option is an integer corresponding to the ASCII character value in decimal. The default is 124, corresponding to the vertical bar or "pipe" character, |.

39.7.5 Alias and address MTA options: `exproute_forward` (0 or 1)

The `exproute_forward` MTA option controls the application of the [exproute channel option](#) to forward-pointing (To:, Cc:, and Bcc: lines) addresses in the message header. A value of 1 is the default and specifies that `exproute` should affect forward-pointing header addresses. A value of 0 disables the action of the `exproute` channel option on forward-pointing addresses.

39.7.6 Alias and address MTA options: `improute_forward` (0 or 1)

The `improute_forward` MTA option controls the application of the [improute channel option](#) to forward-pointing (To:, Cc:, and Bcc: lines) addresses in the message header. A value of 1 is the default and specifies that `improute` should affect forward-pointing header addresses. A value of 0 disables the action of the `improute` channel option on forward-pointing addresses.

39.7.7 Alias and address MTA options: `local_format_restrictions` (bitmask)

The `local_format_restrictions` MTA option affects whether the pipe character (vertical bar) may appear in local (L) channel addresses. It also affects whether filename delivery format, `+filename@local-channel-domain-name`, is permitted. Note that authenticated submissions bypass such restrictions.

In a configuration with [viaaliasrequired](#) set on the local (L) channel, such as in normal Messaging Server MTA configuration, this option is not really relevant: the [viaaliasrequired](#) effect takes precedence in requiring that every local-part correspond to an actual user -- unless the pipe character or a leading plus character were to occur in a user's e-mail address, these syntaxes would inherently not correspond to actual user e-mail addresses and hence not be permitted. But in a different sort of configuration, such as an old PMDF configuration, this option controls whether the special syntaxes are allowed in addresses matching the local (L) channel. The default value is 1, meaning that (in the absence of [viaaliasrequired](#)), pipe characters are disallowed but filename delivery is allowed.

Table 39.5 local_format_restrictions MTA option bit values

Bit	Value	Usage
0	1	Disallow pipes in local addresses
2	4	Disallow files in local addresses

Bit 0 is the least significant bit.

The error, if such a condition is violated (and with [viaaliasrequired](#) *not* set), will be:

```
5.1.3 invalid material in localpart of address: address
and as an SMTP error:
```

```
553 5.1.3 invalid material in localpart of address: address
```

39.7.8 Alias and address MTA options: max_alias_levels (integer)

The [max_alias_levels](#) MTA option controls the degree of indirection allowed in aliases, that is, how deeply aliases may be nested, with one alias referring to another alias, etc. This applies to [direct LDAP alias lookups](#), as well as to traditional (legacy configuration) [alias database](#) and [alias file](#) lookups. The default value is 10. See [Alias recursion and nested list definitions](#) for some additional discussion.

39.7.9 Alias and address MTA options: missing_recipient_group_text (string)

The [missing_recipient_group_text](#) MTA option specifies the phrase to use when generating an empty group construct; that is, the phrase used when [missing_recipient_policy=4](#) (MTA-wide) or [missingrecipientpolicy=4](#) (channel level) is being applied. The default phrase, if this option is not set, is "Recipients not specified".

39.7.10 Alias and address MTA options: missing_recipient_policy (0-6)

[RFC 822](#) (Internet) messages are required to contain a recipient header: a To:, Cc:, or Bcc: header. A message without any such header is illegal according to [RFC 822](#). Nevertheless,

some broken user agents and mailers (*e.g.*, many older versions of sendmail) will emit such illegal (per [RFC 822](#)) messages. Note that [RFC 5322](#), the update to [RFC 822](#), relaxes the [RFC 822](#) requirement and allows submitted messages to be lacking in any recipient header line. However, unless it is certain that *all* the MTAs and MUAs that may ever handle a message in fact conform to [RFC 5322](#) (rather than the older [RFC 822](#)), it is unwise to emit a message lacking all recipient header lines, since the behavior of an [RFC 822](#)-compliant MTA or mail user agent may be undesirable when encountering a message that is, from its point of view, illegal--results may include rejection of such a message, potentially undesired exposure of recipient information such as recipients intended as Bcc: recipients, *etc.*

The `missing_recipient_policy` MTA option takes an integer value specifying what approach to use for such messages; the default value, if the option is not explicitly present, is 0. The meaning of this default value of 0 has changed: prior to JES MS 6.2, it was equivalent to 2, meaning that envelope To addresses are placed in a To: header line. As of JES MS 6.2, it is equivalent to 1, meaning to pass such messages through unchanged, in accordance with what [RFC 5322](#) now recommends.

Table 39.6 missing_recipient_policy option values

Value	Action
0	Default; in JES MS 6.0 and 6.1 this corresponded to a value of 2, namely place envelope To recipients in a To: header line; as of 6.2, this corresponds to a value of 1, namely pass the message through unchanged, in accordance with what RFC 5322 now recommends.
1	Pass the illegal-per- RFC 822 (though legal per RFC 5322) message through unchanged.
2	Place envelope To recipients in a To: header line.
3	Place all envelope To recipients in a single Bcc: header line.
4	Generate an empty group construct (<i>i.e.</i> , ;) To: header line. The phrase used in the group construct is controlled by the <code>missing_recipient_group_text</code> MTA option, so for instance "To: Recipients not specified: ;".
5	Generate a blank Bcc: header line.
6	Reject the message. The SMTP error issued with such a rejection will be: "554 5.6.0 Error writing message - message is missing required recipient header fields"

Note that the `missingrecipientpolicy` channel option can be used to set per-channel controls for this sort of behavior; such per-channel controls override the setting of the MTA option `missing_recipient_policy`.

39.7.11 Alias and address MTA options: `name_table_name` (string; OpenVMS only)

OpenVMS only.

The `name_table_name` MTA option specifies the name of a logical name table to be searched for address aliases by the MTA. This table name may itself be a logical name (in the process or system directory) which specifies one or more tables to search. This option has no default; if it is not specified logical name tables are not searched for aliases.

39.7.12 Alias and address MTA options: reverse_envelope (0 or 1)

RESTRICTED.

The `reverse_envelope` option controls whether or not the MTA applies address reversal to envelope From addresses as well as header addresses. This option will have no effect unless address reversal is being performed. That is, in order for `reverse_envelope` to have any effect, a `reverse_url` must be set, a [REVERSE mapping](#) must exist, or `use_reverse_database` must be set to a value causing use of the reverse database.

The default for `reverse_envelope` is 1, which means that the MTA will attempt to apply any address reversal to envelope From addresses. A value of 0 will disable address reversal from applying to envelope From addresses; that is, disable the `reverse_url` MTA option setting, the [address reversal database](#), and the [REVERSE mapping](#) from affecting envelope From addresses.

Note that at typical Oracle Messaging Server sites, `reverse_envelope=0` **should not** be set as disabling `reverse_url` lookups on envelope From addresses will disable other intended functionality; see [Intended side effects of LDAP address reversal](#).

39.7.13 Alias and address MTA options: subaddress_char (list of integers)

The `subaddress_char` MTA option specifies the ASCII representation of the character to use as the [subaddress indicator](#) in the mailbox portion of an address. The default is to use the plus character, +, which has ASCII representation 43.

Note that internally the `ims-ms channel` always expects to see a plus character, +, and is normally always passed the plus character due to the [delivery_options](#) MTA option, regardless of what character is used "externally" as the subaddress separator character.

39.7.14 SRS MTA options: token_char (integer position of ASCII character)

RESTRICTED.

The `token_char` MTA option controls what character represents a token in the local-part of addresses. This is relevant for [SRS address handling](#). The value of this option is an integer corresponding to the ASCII character value in decimal. The default is 61, corresponding to the equal sign, =.

39.7.15 Alias and address MTA options: use_alias_database (0 or 1)

The `use_alias_database` MTA option controls whether or not the MTA makes use of the [alias database](#) as a source of system aliases for local addresses. The default is 1, which means that the MTA will check the database if it exists. A value of 0 will disable this use of the alias database.

39.7.16 Alias and address MTA options: use_domain_database (0 or 1)

The `use_domain_database` MTA option controls whether or not the MTA makes use of the [domain database](#) as a source of rewrite rules. In Messaging Server 7.2 and earlier, the default was 1, which means that the MTA would check the database if it existed; as of Messaging Server 7.3, the default is 0, which disables consultation of the domain database.

39.7.17 Alias and address MTA options: use_forward_database (bitmask)

The `use_forward_database` MTA option controls whether or not the MTA makes use of the forward database, and also controls the exact format of probes of the forward database and [FORWARD mapping table](#). The value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 39.7 use_forward_database MTA option bits

Bit	Value	Usage
0	1	When set, the forward database is used.
3	8	When set, channel-level granularity is used with the forward database entries. Forward database entries' left hand sides must have the form (note the vertical bars,) <i>source-channel from-address to-address</i> Note that source-specific forwarding is very seldom appropriate, and in those rare cases where it is appropriate, the forward database is seldom the most suitable choice for achieving it.
4	16	When set, channel-level granularity is used with the FORWARD or any domain catchall mapping . The mapping entries' patterns (left hand sides) must have the form (note the vertical bars,) <i>source-channel from-address to-address</i> Note that source-specific forwarding is very seldom appropriate, and in those rare cases where it is appropriate, the FORWARD mapping is seldom the most suitable choice for achieving it.
5	32	When set, modifies the effect of bit 3 (source-specific forward database probes) by also including the destination channel in the probe.
6	64	When set, modifies the effect of bit 4 (source-specific FORWARD or domain catchall mapping probes) by also including the destination channel in the probe.
7	128	(New in 8.0) When set, includes the initial address presented for alias processing in the FORWARD mapping probe. This address appears immediately before the intermediate address included by bit 8 below.
8	256	(New in 8.0) When set, include the current intermediate address in the FORWARD mapping probe. This address appears immediately before the final recipient address.

use_personal_aliases MTA option

9	512	(New in 8.0) When set, include the authenticated sender address address in the FORWARD or any domain catchall mapping probe. This address appears immediately after the destination channel and before conversion tags.
---	-----	---

Bit 0 is the least significant bit.

The default value for use_forward_database is 0, which means that the MTA will not use the forward database at all. Note that a FORWARD mapping table, if present, is always consulted.

39.7.18 Alias and address MTA options: use_personal_aliases (0 or 1)

The use_personal_aliases MTA option controls whether or not the MTA makes use of personal alias databases as a source of aliases for local addresses. The default is 1, which means that the MTA will check such databases, if they exist. A value of 0 will disable personal aliases and make them unavailable to all users.

39.7.19 Alias and address MTA options: use_reverse_database (bitmask)

The use_reverse_database MTA option controls whether or not the MTA makes use of the [address reversal database](#) and [REVERSE mapping table](#) as a source of substitution addresses. (Note that it can not disable use of any [reverse_url](#) setting, although its bit 2, value 4, does affect the scope of reverse_url application.) Its value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 39.8 use_reverse_database MTA option bits

Bit	Value	Usage
0	1	When set, reverse database based address reversal is applied to addresses after they have been rewritten by the MTA address rewriting process; (this does not affect reverse_url or REVERSE mapping table based address reversal subsequent to address rewriting, which is controlled merely by the existence of a reverse_url setting or existence of a REVERSE mapping table, respectively).+
1	2	When set, reverse database and/or REVERSE mapping based address reversal is applied before addresses have had MTA address rewriting applied to them; (this does not affect reverse_url based address reversal, which always occurs after rewriting has been applied; for the REVERSE mapping, setting this bit causes an additional consultation of the REVERSE mapping prior to address rewriting, in addition to the subsequent to rewriting consultation which will always be performed).+
2	4	When set, address reversal, including the reverse_url option setting if applicable, will be applied to all (except envelope To) addresses, including forward-pointing header addresses, not just to backward-pointing addresses.
3	8	When set, channel-level granularity is used with the REVERSE mapping . REVERSE mapping table (pattern) entries must have the form (note the vertical bars,) <i>source-channel destination-channel address</i>

4	16	When set, channel-level granularity is used with address reversal database entries. Reversal database entries' left hand sides must have the form (note the vertical bars,) <i>source-channel destination-channel address</i>
5	32	Apply REVERSE mapping even if a reverse database entry has already matched.
6	64	Apply address reversal to message ids; see Internal host names in Received: and Message-Id: header lines for an example.
7	128	When set, this modifies the effect of bit 4 (channel-level granularity of address reversal database entries); when this bit is also set, the address reversal database entries take the form (note the vertical bars,) <i>destination-channel address</i>
8	256	When set, this modifies the effect of bit 3 (channel-level granularity of REVERSE mapping table entries); when this bit is also set, the REVERSE mapping table entries take the form (note the vertical bars,) <i>destination-channel address</i>
10	1024	During subsequent-to-rewriting address reversal, (that is, that reversal due to having bit 0 (value 1) set), this disables the normal initial reverse database lookup, though a database lookup can still be caused by a \$D in the REVERSE mapping table
11	2048	During prior-to-rewriting address reversal, (that is, that reversal due to having bit 1 (value 2) set), this disables the normal initial reverse database lookup, though a database lookup can still be caused by a \$D in the REVERSE mapping table
12	4096	(New in 8.0.) When set, include the name of the header field the address being processed came from in the mapping probe, delimited by vertical bars, immediately after source channel, destination channel, and conversion tag information. A trailing colon is always included in the field name. A blank name appears when envelope addresses are being processed.
13	8192	(New in 8.0.) When set, do not impose source block and recipient limits and capture actions based on the envelope from (MAIL FROM) address. For background discussion, see Intended side effects of LDAP address reversal .
14	16384	(New in 8.0.) When set, do not impose source block and recipient limits and capture actions based on the authenticated sender address. For background discussion, see Intended side effects of LDAP address reversal .

+In initial iMS 5.2 and earlier versions, the 0th and 1st bits of use_reverse_database not only controlled when, but also *whether* a [REVERSE mapping table](#) would be consulted at all for address rewriting; now if a REVERSE mapping table exists, it definitely *will* be consulted for address reversal (at least) subsequent to address rewriting; (depending upon bit 1, it may also be consulted prior to address rewriting). So this is a change from iMS 5.2 and earlier versions.

Bit 0 is the least significant bit.

The default value for use_reverse_database is 5, which means that in addition to consulting any [reverse_url](#) setting to reverse envelope From addresses and both backwards and forwards pointing header addresses after they have passed through the normal address rewriting process, the MTA will also consult any [reverse database](#) or [REVERSE mapping](#) to reverse Envelope From addresses and both backwards and forwards pointing header

addresses after they have passed through the normal address rewriting process. Simple address strings are presented to both the REVERSE mapping and the reverse database. Note that a value of 0 disables the use of the address reversal completely. (Note that the default of 5 represents a change from early versions of PMDF in which this option had a default value of 1 (reverse only backwards pointing addresses).)

39.7.20 Alias and address MTA options: user_case (0 or 1)

The `user_case` MTA option causes some effects of forcing user names to lower case (on certain "local" sorts of channels), affects the forcing-to-lowercase of variants of "postmaster", and can influence the sorting of addresses. The default is 1, which has the general meaning that case sensitive user names are permitted, except that any "postmaster" variants are forced to lower case. If set to 0, and if the `localbehavior` channel option is set either explicitly on a channel (or implicitly forced as with the `l` channel), then general user names matching that channel will be forced to lower case.

The `user_case` MTA option also potentially, depending upon other conditions, may affect the sorting of addresses.

39.8 Autoresponse periodicity MTA options

The MTA has several options affecting its memory/tracking of previous vacation (autoreply) messages, and as of 7.0.5 its [Sieve notify autoresponses](#), and hence its limiting of successive such messages: in other words, options controlling the periodicity of autoresponses.

For each user, the MTA maintains an autoresponse information file, the purpose of which is to record for which prior messages a vacation message has already been sent, (or as of 7.0.5, a Sieve `notify` action was performed), used to avoid sending additional identical autoresponse messages "too soon". The per-user, autoresponse information files are flat text files, one per local user. The files are located and named via a configurable template, specified by the `vacation_template` MTA option. In a user's autoresponse file, the MTA records the most recent time at which the user sent back what would be the "same" vacation message to the "same" original sender (same recipient of potential same vacation message). And as of 7.0.5, the MTA similarly records in the autoresponse file the most recent time the user had a [Sieve notify action](#) performed in response to the "same" message sender.

An explicit [vacation action](#) in a Sieve script may specify the timeout (period between sending back another "identical" vacation message to the same sender) via the standard `:days` parameter, or the MTA's extension `:hours` or `:seconds` parameters. Or a user's `mailAutoReply*` LDAP attribute values (which the MTA uses to construct on-the-fly a Sieve vacation action) may specify such a timeout via the value of the LDAP attribute named by the `ldap_autoreply_timeout` MTA option (default `mailAutoReplyTimeout`); if the user does not have their own `mailAutoReplyTimeout` set, then the domain value, if specified via the LDAP attribute named by the `ldap_domain_attr_autoreply_timeout` MTA option, will be used; or if neither the user nor the domain has such a value set, then the system wide default specified via the `autoreply_timeout_default` MTA option will be used.

Any such timeout settings must be greater than or equal to the minimum allowed by the `vacation_minimum_timeout` MTA option; values less than that will be silently (no error) adjusted upwards to the `vacation_minimum_timeout` value. Similarly, as of the 8.0 release, values greater than the value of the `vacation_maximum_timeout` MTA option will be silently adjusted downwards to that option's value.

When the MTA is deciding whether or not to generate a vacation message back to some original sender due to existence of either an explicit `vacation` action in a Sieve applying for the original message recipient, or `mailAutoReply*` LDAP attributes (more precisely, LDAP attributes named by `ldap_autoreply_*` MTA options) of the original message recipient, the MTA looks in the autoreponse suppression file corresponding to the original message recipient, looking up with a key based on the original message sender and the substance of the original message recipient's vacation message to find the time (if any) of the last such vacation response. If the time is "too soon", no vacation message will be generated. Here "too soon" is as defined above: set either via an explicit `vacation` action `timeout` parameter, or `mailAutoReplyTimeout` if a `vacation` action is being generated for a user due to `mailAutoReply*` attribute use, as defaulted by domain and system defaults, and constrained by the system minimum permissible value.

A user's `notify` Sieve action may specify an explicit timeout via use of a `:days`, `:hours`, or `:seconds` parameter. Any such value must be at least the value of the `notify_minimum_timeout` MTA option and no greater than the value of the `notify_maximum_timeout` MTA option, or it will be silently (with no error) adjusted to conform to the permitted range. If no explicit timeout is specified, then the value of the `notify_timeout_default` MTA option is used.

Maintenance (clean up) of the per-user autoreponse files is performed automatically by the MTA, on a lazy (only when a file is already being opened), randomized (not performed every time) basis. In particular, the `vacation_cleanup` MTA option controls the probabilistic frequency of clean up of expired old entries from any such file.

For options relating to other aspects of vacation messages, see also the `max_vacations` MTA option and all the rest of the MTA options naming various LDAP attributes that specify aspects of vacation messages, `ldap_start_date`, `ldap_end_date`, `ldap_autoreply_mode`, `ldap_autoreply_subject`, `ldap_autoreply_text`, `ldap_autoreply_text_internal`, `ldap_autoreply_addresses`, as well as the already mentioned `ldap_autoreply_timeout`, and `ldap_domain_attr_autoreply_timeout`.

39.8.1 Autoreponse periodicity MTA options: `autoreply_timeout_default` (non-negative integer)

The `autoreply_timeout_default` MTA option specifies the default duration, in hours, for successive vacation (autoreply) responses to any given sender. This system-wide value specifies a default both for `vacation` actions specified explicitly in Sieve filters, and for implicit "vacation" effects resulting from LDAP `mailAutoReply*` attributes.

This system-wide default may be overridden in explicit "`vacation`" actions in Sieve filters by specifying a desired timeout via a `:days`, `:hours`, or `:seconds` parameter. Or for implicit "vacation" actions generated due to LDAP `mailAutoReply*` attributes, this system-wide default may be overridden on a per-domain basis via the domain attribute named by the `ldap_domain_attr_autoreply_timeout` MTA option, or on a per-user basis via the user attribute (normally `mailAutoReplyTimeout`) named by the `ldap_autoreply_timeout` MTA option. The default for this option is 168 (*i.e.*, 7×24), meaning that vacation messages would only be sent back to a given sender once a week (no matter how many messages the sender sends).

39.8.2 Autoreponse periodicity MTA options: `notify_maximum_timeout` (integer)

notify_minimum_timeout
MTA option

(New in 7.0.5.) The `notify_maximum_timeout` MTA option establishes a maximum value, in seconds, for the Sieve "notify" action's `:days`, `:hours`, and `:seconds` parameters. (`:days` and `:hours` values are converted into seconds for the comparison.) Values higher than the maximum are silently adjusted down to the maximum; no error occurs. The default value for `notify_maximum_timeout` is the maximum allowed integer, $2^{31}-1$.

39.8.3 Autoresponse periodicity MTA options: `notify_minimum_timeout` (integer)

(New in 7.0.5.) The `notify_minimum_timeout` MTA option establishes a minimum value, in seconds, for the Sieve "notify" action's `:days`, `:hours`, and `:seconds` parameters. (`:days` and `:hours` values are converted into seconds for the comparison.) Values lower than the minimum are silently adjusted up to the minimum; no error occurs. The default value for `notify_minimum_timeout` is 0.

39.8.4 Autoresponse periodicity MTA options: `notify_timeout_default` (non-negative integer)

The `notify_timeout_default` MTA option specifies the default timeout, in seconds, for suppression of duplicate notifications to a given recipient. It defaults to one hour for the [old form of notify actions](#) and to two minutes for the [new form](#).

39.8.5 Autoresponse periodicity MTA options: `vacation_cleanup` (non-negative integer)

The `vacation_cleanup` MTA option sets a modulus for the frequency at which per-user per-response vacation files are "cleaned up", (that is, scanned and expired entries removed). Each time one of the per-user per-response vacation files is opened (those files specified via the `vacation_template` MTA option), the value of the current time in seconds modulo the `vacation_cleanup` value is computed. If the result is zero, then the file is scanned and all expired entries are removed. The default value for the option is 200, which means that there is a 1 in 200 chance that a cleanup pass will be performed.

39.8.6 Autoresponse periodicity MTA options: `vacation_maximum_timeout` (integer)

(New in 7.0.5.) The `vacation_maximum_timeout` MTA option establishes a maximum value, in seconds, for the Sieve "vacation", `:days`, `:hours`, and `:seconds` parameters. (`:days` and `:hours` values are converted into seconds for the comparison.) Values higher than the maximum are silently adjusted down to the maximum; no error occurs. The default value for `vacation_maximum_timeout` is the maximum allowed integer, $2^{31}-1$.

Since the value of the `mailAutoReplyTimeOut` LDAP attribute (or more precisely the value of whatever LDAP attribute is named by the `ldap_autoreply_timeout` MTA option) is converted into such a Sieve "vacation" parameter, the `vacation_maximum_timeout` MTA option value can affect the interpretation of any `mailAutoReplyTimeout` values also.

39.8.7 Autoresponse periodicity MTA options: `vacation_minimum_timeout` (integer)

(New in 7.0.5.) The `vacation_minimum_timeout` MTA option establishes a minimum value, in seconds, for the Sieve `"vacation"`, `":days"`, `":hours"`, and `":seconds"` parameters. (`":days"` and `":hours"` values are converted into seconds for the comparison.) Values lower than the minimum are silently adjusted up to the minimum; no error occurs. The default value for `vacation_minimum_timeout` is 0.

Since the value of the `mailAutoReplyTimeOut` LDAP attribute (or more precisely the value of whatever LDAP attribute is named by the `ldap_autoreply_timeout` MTA option) is converted into such a Sieve `"vacation"` parameter, the `vacation_minimum_timeout` MTA option value can affect the interpretation of any `mailAutoReplyTimeout` values also.

39.8.8 Autoresponse periodicity MTA options: `vacation_template` (file or memcache URL)

The `vacation_template` MTA option specifies a template for the name and location of the per-user autoresponse information memcache entries or files. These can be flat text files, one per local user. The value should be a `memcache: URL` or a `file URL` (`file:path-template`).

Various substitution sequences may be used in constructing the file path template; see the MTA's [LDAP URL substitution sequences](#). Note that the `$nA` and `$U` substitution metacharacters are likely to prove particularly useful in constructing effective `vacation_template` settings.

This option must be set to a sensible value in order to support user `"vacation"` Sieve script actions (and LDAP `mailAutoReply*` attributes). Prior to Messaging Server 7.4-0.01, there was no default value. As of Messaging Server 7.4-0.01, the default value is:

```
file:///msg-install-path/data/vacation/$3I/$1U/$2U/$U.vac
```

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 filesystem. Note that when intending to use NFS to share vacation response files among multiple systems, it is important to define the same MTA user account -- same user name and uid (see the `user` option in `restricted.cnf`) -- on each system so as to avoid permission problems.

As of the 8.0 release, memcache is also supported as a back end for vacation timeout information. This is accomplished by specifying a `memcache: URL`. A typical setting would be:

```
memcache:/// $U@$2I
```

in which case the memcache server is specified by the `memcache_host` and `memcache_port` MTA options. These options can be overridden by specifying the host and port in the URL, e.g.,

```
memcache://host:port/$U@$2I
```

The content of the URL after the host and port specifies the key used to store information in memcache. The `$U` substitution provides the necessary information for the key, but a prefix or suffix can also be included if there is a need to distinguish the keys from other information stored in the memcache instance.

39.9 BURL MTA options

The BURL extension to SMTP SUBMIT is defined in [RFC 4468 \(Message Submission BURL Extension\)](#). The MTA's [SMTP SUBMIT server](#) can support this extension, if configured to do so.

Configuration of BURL support involves setting the two MTA options [imap_username](#) and [imap_password](#), as well as configuring the [BURL_ACCESS mapping table](#). The BURL MTA options [imap_username](#) and [imap_password](#) specify the credentials for the MTA to use when it connects to the IMAP server as the "submit" user, so of course these credentials (these MTA option settings) must match those configured for the IMAP server's "submit" user as configured via the [submituser IMAP option](#).

39.9.1 BURL MTA options: [imap_password](#) (string)

In order to perform an IMAP BURL operation, the [SMTP SUBMIT server](#) has to have the ability to log in to the IMAP server as the submit user. The [imap_password](#) MTA option specifies the password to use for such operations (and of course must match the password value set for the [submituser account](#)). This option has no default.

39.9.2 BURL MTA options: [imap_username](#) (string)

In order to perform an IMAP BURL operation, the SMTP SUBMIT server has to have the ability to log in to the IMAP server as the submit user. The [imap_username](#) MTA option specifies the submit user; if not set, it defaults to the setting of the [imap.submituser](#) option (corresponding to the old configutil parameter `service.imap.submituser`).

39.10 Configutil override MTA options

Historically, the MTA has had a number of options available to override (specifically for MTA purposes) various general Messaging Server settings formerly set via `configutil`. In Unified Configuration, such former `configutil` options typically are now [base level options](#) -- and MTA-specific overrides via MTA options typically still exist. (A convenient concordance of many such base, `configutil`, and MTA options may be found in [Basic configuration settings relevant to alias LDAP lookups](#).)

See in particular the options:

- [ldap_default_domain](#)
- [ldap_domain_root](#)
- [ldap_host](#)
- [ldap_host_alias_list](#)
- [ldap_local_host](#)
- [ldap_mail_aliases](#)
- [ldap_pab_host](#)
- [ldap_pab_max_connections](#)
- [ldap_pab_password](#)
- [ldap_pab_port](#)
- [ldap_pab_username](#)
- [ldap_password](#)
- [ldap_port](#)

- [ldap_schematag](#)
- [ldap_username](#)
- [ldap_user_root](#)
- [projectid](#)

39.11 Conversions MTA options

The MTA has a number of options relating to conversion operations. The most fundamental of these in Unified Configuration is [conversions](#), which replaces the legacy configuration `conversions` file, being where conversions entries are stored. Additional options relating to conversion operations include:

- [conversion_size](#), [personal_conversion_size](#), [string_pool_size_0](#), and [string_pool_size_4](#), which respectively control the maximum number of conversion entries, the maximum number of personal conversion entries, and limits the total characters in conversions entries and in personal conversions entries.
- [include_conversiontag](#) and [original_channel_probe](#), which affect what information is used in certain [mapping table](#) probes; and
- [log_conversion_tag](#), which controls whether or not message [conversion tags](#) are included in [MTA message transaction log entries](#).

39.11.1 conversions Option

The `conversions` MTA option stores all conversion entries. It corresponds to the legacy configuration `conversions` file.

The `conversions` MTA option would typically be set or modified by using the `edit` command of `msconfig`, *e.g.*:

```
msconfig> edit conversions
```

39.12 Counters MTA options

The MTA has a number of options relating to its counters.

The [circuitcheck_completed_bins](#) MTA option relates to MTA circuit check counter binning. The [log_delay_bins](#) and [log_size_bins](#) MTA options relate to MTA counters binning. See also the [sndopr_priority](#) MTA option, which controls the syslog facility and severity of, among other things, syslog notices generated if and when the MTA encounters trouble with its association counters.

The [log_debug](#) MTA option enables low-level debugging (typically only meaningful to Oracle support) regarding the MTA's transaction logging and the incrementing of MTA channel counters.

39.12.1 Counters MTA options: `circuitcheck_completed_bins` (comma-separated list of up to eight integers)

The `circuitcheck_completed_bins` MTA option specifies the bin divisions, in seconds, for MTA circuit check counters. It takes as argument a list of up to eight integer values. The default values are 120, 300, 900, 1800, 3600, 7200, 14400, and 28800; *i.e.*, two minutes, five minutes, fifteen minutes, thirty minutes, one hour, two hours, four hours, and eight hours, respectively.

39.12.2 Counters MTA options: `enable_delay_timers` (0 or 1)

The MTA can optionally maintain timers to measure delays caused by various external processes. Setting the `enable_delay_timers` MTA option to a value of 1 enables these timers. A value of 0 (the default) disables them.

Timings are done on a per-message basis, if enabled. For details on what timers are available, see the `log_callout_delays` MTA option which discusses the details of the timers, as well as the format in which they may optionally be logged.

39.12.3 Counters MTA options: `log_delay_bins` (comma-separated list of up to five integers)

The `log_delay_bins` MTA option specifies the bin divisions for the [MTA counters](#) tracking numbers of messages delivered in the specified number of seconds. The default values are 60, 600, 6000, 60000, 600000.

39.12.4 Counters MTA options: `log_frustration_limit` (integer)

When attempting a [counter](#) update operation (whether an attempt to update channel counters or association counters), the MTA will try ten times before giving up on that particular update operation. Each MTA process keeps track of how many such update "frustrations" have occurred, that is, how many times the process has had to give up on updating the counters, and if that value exceeds the `log_frustration_limit` value, which by default is 100, then that process will no longer even attempt counter update operations.

39.12.5 Counters MTA options: `log_size_bins` (comma-separated list of up to five integers)

The `log_size_bins` MTA option specifies the bin divisions for the [MTA counters](#) tracking numbers of messages of the specified number of (MTA) blocks (the size of an MTA block having been defined via the `block_size` MTA option). The default values are 2, 10, 50, 100, 500.

39.12.6 Counters MTA options: `log_statistics` (0, 1, or 2)

Usage: RESTRICTED. (In PMDF V5.1 and earlier, `log_statistics=0` told PMDF not to generate and maintain its [counters](#); but as of PMDF V5.2 and for all versions of

the Messaging Server MTA, `log_statistics=0` has no effect and is equivalent to `log_statistics=1`.) `log_statistics=1` is the default and means to create counters normally. `log_statistics=2` causes "strict" creation of counters; the MTA aborts if the counters cannot be created.

39.13 Database MTA options

The MTA has a number of options relating to its use of databases. These include:

- [alias_magic](#) controlling the order of consulting various sources (including databases) for aliases;
- [forward_data_size](#), [general_data_size](#), and [reverse_data_size](#) controlling the internal size the MTA allots for its "in-memory" versions of the forward database, general database, and reverse database, when use of such "in-memory" databases has been selected via [use_text_databases](#);
- [name_table_name](#) (OpenVMS only) which specifies a local name table used to store aliases;
- [queue_cache_mode](#) which specifies use of an "in-memory" queue cache database by the [Job Controller](#);
- [use_alias_database](#), [use_domain_database](#), [use_forward_database](#), [use_personal_aliases](#), and [use_reverse_database](#) which, among other effects, control the use and format of various databases; and
- [vacation_template](#) specifies the location of the "database" of per-user vacation response data;
- (new in Messaging Server 8.0) [alias_database_url](#), [domain_database_url](#), [forward_database_url](#), [general_database_url](#), [reverse_database_url](#), and [ssr_database_url](#) which specify [Memcache URLs](#) for storing MTA database data.

See also the [Direct LDAP MTA options](#), the [Autoresponse periodicity MTA options](#), and the [MeterMaid MTA options](#).

39.14 Debug MTA options

Debugging of various general MTA facilities and components may be enabled via several options. Note the distinction between debugging (verbose output especially focused on problem detection) *vs.* [transaction logging](#) (recording of processing) *vs.* event logging (call-out logging of events of particular interest).

These MTA-wide facility debug options enable certain sorts of (generally MTA low level) debugging irrespective of what MTA component is operating. For component specific debugging, including "higher level" debugging of that component's specific operation, see also component level debugging such as the channel level [master_debug](#) and [slave_debug](#) channel options, or the [debug](#) Job Controller or Dispatcher option.

For a quick view of whether/what debugging you may have enabled, try:

```
msconfig> show *debug
```

39.14.1 loglevel Option

The `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information`, or `debug`. This is ignored by the [Dispatcher](#) and [Job Controller](#) unless the `use_nslog` option is set for them, and ignored by the MMP if `use_nslog` is set to 0.

39.14.1.1 Use with ens

The `loglevel` ENS option specifies the level of ENS client library logging to the process `nslog` file. Messages are also filtered per the `loglevel` of the process. Allowed values are as in `logfile.*.loglevel`. But the value "debug" generates lots of data and is not recommended.

39.14.1.2 Use with messagetrace

Message Trace data is inherently `information` level. So setting the `messagetrace.loglevel` option to a higher value than `information` will suppress the recording of Message Trace data.

39.14.1.3 Use with mta

The `loglevel` option for the MTA specifies an MTA log level used for the `imta` log file (primarily used by [ims_master](#) and the [LMTP server](#)). Its value can be one of `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information`, or `debug`.

Note that this MTA-wide setting of `mta.loglevel` (by default affecting both `ims-ms` channels and the [LMTP server](#)) can be overridden for the [LMTP server](#) via a `tcp_lmtp_server.loglevel` setting.

39.14.1.4 Use with the MMP

The MMP's `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than 'error'. For backwards compatibility, MMP configuration files may use integer settings from 3 to 7 for 'error' to 'debug' respectively, or 0 for `nolog`.

39.14.1.5 Use with the IMAP proxy

The IMAP Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than 'error'.

39.14.1.6 Use with the POP proxy

The POP Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than 'error'.

39.14.1.7 Use with the submit proxy

The SUBMIT Proxy `loglevel` option can be: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information` or `debug`. The MMP will not generate messages with priority higher than 'error'.

39.14.2 Debug MTA options: `ap_debug` (integer)

RESTRICTED. The `ap_debug` MTA option enables internal MTA debugging; it is intended for use by Oracle and is not intended to be used by end sites. Specifically, it enables debugging of the AP routines (low level address parsing routines). Higher settings of `ap_debug` cause more verbose output. The precise debug output can be expected to vary between and during releases. When enqueueing messages, in order for `ap_debug` values greater than 0 to take effect, both the `master_debug` and `slave_debug` channel options must be set on the source channel except for the L channel where just `master_debug` or `slave_debug` is sufficient.

39.14.3 Debug MTA options: `cache_debug` (0 or 1)

The `cache_debug` MTA option controls debugging regarding the direct LDAP caching of lookup results (domains, aliases, address reversals). The default is 0, meaning that such debug output is disabled. Setting this option to 1 enables the debug output; the information is output just before the MTA component exits. (So for instance, `imsimta test -rewrite` will output information regarding what it cached at the end of its other, normal output--since `imsimta test -rewrite` most often is used to test just one or a couple of addresses, there is not often much it had to cache. An SMTP server process will output to its log file information regarding what it cached just before it shuts down at the end of its lifetime -- since SMTP server processes typically last for some time, there may be quite a bit of caching that occurred.)

39.14.4 Debug MTA options: `config_debug` (integer)

RESTRICTED. The `config_debug` MTA option enables internal MTA debugging; it is intended for use by Oracle and is not intended to be used by end sites.

39.14.5 Debug MTA options: `debug_flush`

As of Messaging Server 7.1, a.k.a. Messaging Server 7.0-3.01, the `debug_flush` MTA option causes certain debug output to get immediately flushed to disk. This is applicable for many MTA components, including typical [channel debug output](#), but it is especially relevant and noticeable for long-running components such as the [SMTP server](#), and [Job Controller](#). The flush-to-disk-log-file of the debug output may incur a bit of a performance penalty, but tends to be more convenient for debugging purposes. The default is that such debug flushing is not enabled (`debug_flush = 0`).

As of 7.0.5, the `debug_flush` MTA option can also cause flushing of [Dispatcherdebug](#) output.

39.14.6 Debug MTA options: `dequeue_debug` (0 or 1)

The `dequeue_debug` MTA option specifies whether or not debugging output from the MTA's dequeue facility QU is produced; (note that master channel programs, since they are generally dequeue oriented, usually use QU routines). If enabled with a value of 1, this output will be produced on all channels that use the QU routines. The default value of 0 disables this output.

39.14.7 Debug MTA options: filter_debug (0 or 1)

New in Messaging Server 6.2. Control whether the stack state information part of [Sieve filter](#) debugging is put in debug logs. (Note that this is quite "low level" debugging, not likely to be of interest unless requested by Oracle support.)

For debugging of Sieve filters, see also the [imsimta test -expression utility](#) and the [mm_debug](#) MTA option along with the [Sieve debug action](#).

39.14.8 Debug MTA options: log_debug (0 or 1)

RESTRICTED: The `log_debug` MTA option may be used to enable internal debugging of MTA logging activity, including MTA transaction logging and incrementing of the MTA channel counters; it is intended for use by Oracle, and is not intended to be used by end sites.

39.14.9 Debug MTA options: mm_debug (integer)

RESTRICTED: The `mm_debug` MTA option enables internal enqueue debugging; it is intended for use by Oracle, and is not intended to be used by end sites. In particular, it is not intended for, and is not supported for use for, logging purposes. Specifically, it enables debugging of the MM routines (enqueue routines handling address rewriting, mappings conversions, *etc.*). Note that generally at least one of [master_debug](#) or [slave_debug](#) must be set on a channel in order for `mm_debug` settings to take effect; the OpenVMS L channel is an exception and requires setting both `master_debug` and `slave_debug`. (In PMDF V5.1 and earlier, all channels required setting both `master_debug` and `slave_debug` for `mm_debug` to take effect.)

Higher settings of `mm_debug` cause more verbose output. For instance, currently a value of 1 or more includes some message file access debugging and some [alias file/database](#) access debugging and some overview of header processing and debugging of [access mapping table checks](#) and debugging of [address/subaddress variant](#) lookups, and some commenting about [spam/virus filter package verdicts](#), and some debugging of [SPF lookups checked at MAIL FROM or RCPT TO](#) stages; a value of 2 or more includes debugging of [conversion probes](#) and debugging of [Sieve filter access](#) and some debugging of URL lookups (LDAP and file) and debugging of auto-registration and debugging of [mailing list keyword](#) processing and debugging of [constructing address/subaddress variants](#) and debugging of domainMap/domain-match-cache usage and debugging of whether TLS use is attempted, and further information regarding spam/virus filter package access and errors; a value of 3 or more includes debugging of [mapping table](#) use and further debugging of the actual attribute lookups for LDAP URL lookups and domain lookups (including domain caching info) and (as of JES MS 6.2) debugging regarding the reason when a message for multiple recipients is "split up" into different copies; a value of 4 or more includes some file reading debug especially on OpenVMS; a value of 5 causes very verbose output, including actual contents of the incoming message body, and details regarding spam/virus filter package opt in or opt out; in particular, a value of 8 or more includes some debugging of creating message files and writing out lines of message files and reading in message lines.

The precise debug output resulting from `mm_debug` can be expected to vary between and during releases.

39.14.10 Debug MTA options: os_debug (0 or 1)

RESTRICTED. This `os_debug` MTA option enables internal MTA debugging; it is intended for use by Oracle and is not intended to be used by end sites. Specifically, it enables debugging of the OS routines (low level operating system interface routines), including some file handling and date/time operation routines. The precise debug output can be expected to vary between and during releases.

The `dequeue_debug` MTA option must be enabled also, in order for `os_debug=1` to take effect when dequeuing messages. When enqueueing messages, in order for `os_debug=1` to take effect, both `master_debug` and `slave_debug` must be on the channel. (The [PORT_ACCESS mapping table \\$U flag](#) may be used to selectively enable `slave_debug` on incoming connections.)

As of Messaging Server 7.0.5, enabling `os_debug` will cause the MTA to pay attention to a `debugkeys` Base option value of `lpool`, (`local.debugkeys lpool` in legacy configuration), and thus generate `lpool` debug output.

39.14.11 Debug MTA options: `post_debug` (0 or 1)

DEPRECATED.

Formerly, for PMDF prior to the enhanced Job Controller, the `post_debug` MTA option specified whether or not debugging output was produced by PMDF's periodic delivery job. (No such periodic delivery job exists for the Messaging Server MTA; instead, the Job Controller handles scheduling and initiating delivery re-try jobs as appropriate.)

39.14.12 Debug MTA options: `return_debug` (0 or 1)

The `return_debug` MTA option enables or disables debugging output in the nightly message bouncer job (the return job). A value of 0 disables this output (the default) while a value of 1 enables it. Debugging output, if enabled, in iMS 5.2 will appear in the [Job Controller's](#) log file, usually a `job_controller.log-*` file, and in JES MS 6.1p1 and later will appear in the `IMTA_LOG:return-uniquestring.log` log file.

39.14.13 `return_verify` Option

The `return_verify` MTA option was introduced in MS 7.0.5, in place of the former MTA Tailor option `imta_return_verify`. When the `return_verify` MTA option is set to 1, shell script logging, `-x`, will be enabled within the `return_job` shell script.

39.14.14 `symbiont_debug` Option

DEPRECATED.

Formerly, the `symbiont_debug` MTA option could be used to enable debugging of the PMDF Process Symbiont.

39.14.15 Debug MTA options: `tracking_debug` (0-10)

RESTRICTED: The `tracking_debug` MTA option is used to enable debug output from the MTA's tracking subsystem. The default value of 0 disables debug output; positive values enable it. The larger the value, the more output is produced.

39.15 Direct LDAP MTA options

In modern configurations, provisioning of mail domains, and provisioning of users, mail groups, and mail lists -- [aliases](#) from the MTA's point of view -- is typically done in LDAP. This is sometimes referred to as "[Direct LDAP](#)" provisioning or "[Direct LDAP](#)" aliases, in contrast to the older style of having MTA rewrite rules keep track of "local" domains, and storing [aliases](#) for users in those domains in the MTA [alias file](#) or MTA [alias database](#) and the MTA [reverse database](#).

There are many MTA options for controlling the many aspects of so-called "Direct LDAP" domain and alias lookups, that range from those controlling the [basics of connecting to LDAP](#), to [basics of the LDAP schema and DIT layout](#), to [tweaking the interpretation of LDAP attributes](#), to [specifying the names of the LDAP attributes of interest](#) (re-vectoring LDAP attribute names to allow use of any semantically-compatible schema), including some [attributes fetched upon successful authentication](#), to details of [looking up domains in LDAP](#), then in such domains details of [looking up users in LDAP](#), and finally [caching LDAP lookup results](#).

39.15.1 LDAP bind and connect MTA options

MTA options exist to control the basics of its LDAP bind operations and LDAP connections: the credentials used to bind, the timeout on connection attempts, the maximum number of simultaneous connections, *etc.* Besides these MTA options controlling the MTA's "normal" LDAP connections (those LDAP connections used for lookups in the domain and user/group portions of the DIT, as well as general LDAP URL lookups sites have explicitly configured in MTA rewrite rules or mapping tables), the MTA is also capable of performing PAB (Personal Addressbook) specific LDAP lookups (see the [LDAP PAB MTA options](#)) as well as LDAP connections to some alternate, "external" LDAP directory (see the [LDAP external directory lookup MTA options](#)).

39.15.1.1 LDAP bind and connect MTA options: `ldap_host` (host)

The `ldap_host` MTA option specifies the default host to which to connect when making LDAP queries. This option, if set, overrides for MTA purposes the [ugldaphost](#) base-level option (in legacy configuration, the `local.ugldaphost` configutil parameter).

Prior to Messaging Server 7.0u4, the MTA's LDAP queries -- that is, [ldap: URL](#) uses in MTA rewrite rules, mapping tables, alias translations value, *etc.* -- in fact did not allow specifying a host in the URL itself and instead *required* that a default LDAP host had to have been specified via `ldap_host` or [ugldaphost](#).

39.15.1.2 LDAP bind and connect MTA options: `ldap_max_connections` (non-negative integer)

The `ldap_max_connections` MTA option takes a non-negative integer argument specifying the maximum number of simultaneous LDAP connections that the MTA can use. The default is 1024.

39.15.1.3 LDAP bind and connect MTA options: `ldap_password` (string)

The `ldap_password` MTA option specifies the password to use when binding for LDAP queries. If set, this option overrides the `local.ugldapbindcred` configutil parameter in legacy configuration; in Unified Configuration, the equivalent option is `ugldapbindcred`.

39.15.1.4 LDAP bind and connect MTA options: `ldap_port` (integer)

The `ldap_port` MTA option specifies the port to which to connect when making LDAP queries. If set, this option overrides the `ugldapport` base option (formerly the `local.ugldapport` configutil parameter).

Prior to Messaging Server 7.0u4, the MTA's LDAP queries -- that is, `ldap: URL` uses in MTA rewrite rules, mapping tables, alias translations value, *etc.* -- in fact did not allow specifying a port in the URL itself and instead *required* that a default LDAP port had to have been specified via `ldap_port` or `ugldapport`.

39.15.1.5 LDAP bind and connect MTA options: `ldap_timeout` (integer)

The `ldap_timeout` MTA option controls how long to wait (in hundredths of seconds) before timing out on an LDAP query. The default value is 180000. Note that some underlying or shared code may have its own, separately controlled LDAP timeout settings. In particular, some code may instead use the general `ldapsearchtimeout` base option (formerly the `local.ldapsearchtimeout` configutil parameter setting), whilst other code uses the `ADMLDAP_TIMEOUT` environment variable setting (and defaults to 60 seconds if that variable is not set).

39.15.1.6 LDAP bind and connect MTA options: `ldap_use_async` (bitmask)

The `ldap_use_async` MTA option controls the use of asynchronous (as opposed to synchronous) LDAP lookups. Asynchronous lookups avoid the need to store an entire large LDAP result in memory, which seems to cause performance problems in some cases. This option takes 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. The following bits are defined:

Table 39.9 `ldap_use_async` MTA option bits

Bit	Value	Usage
0	1	<code>ldap_group_url1</code> (<code>mgrpDeliverTo</code>) URLs
1	2	<code>ldap_group_url2</code> (<code>memberURL</code>) URLs
2	4	<code>ldap_group_dn</code> (<code>uniqueMember</code>) DN's and as of Messaging Server 7.0.5, also <code>ldap_group_dn2</code> DN's
3	8	[<code>AUTH_LIST</code>], [<code>MODERATOR_LIST</code>], [<code>SASL_AUTH_LIST</code>], [<code>SASL_MODERATOR_LIST</code>] list named parameter URLs (legacy configuration), or in Unified Configuration, <code>alias_auth_list</code> , <code>alias_moderator_list</code> , <code>alias_sasl_auth_list</code> , <code>alias_sasl_moderator_list</code> , alias option URLs
4	16	[<code>CANT_LIST</code>], [<code>SASL_CANT_LIST</code>] list named parameter URLs (legacy configuration), or in Unified Configuration, <code>alias_cant_list</code> , <code>alias_sasl_cant_list</code> alias option URLs

5	32	[ORIGINATOR_REPLY] list named parameter URLs (legacy configuration), or in Unified Configuration, alias_originator_reply alias option URLs
6	64	[DEFERRED_LIST], [DIRECT_LIST], [HOLD_LIST], [NOHOLD_LIST] list named parameter URLs (legacy configuration), or in Unified Configuration, alias_deferred_list , alias_direct_list , alias_hold_list , alias_nohold_list alias option URLs
7	128	[USERNAME_AUTH_LIST], [USERNAME_MODERATOR_LIST], [USERNAME_CANT_LIST] list named parameter URLs (legacy configuration), or in Unified Configuration, alias_username_auth_list , alias_username_moderator_list , alias_username_cant_list alias option URLs
8	256	alias file list URLs
9	512	alias database list URLs
10	1024	ldap_cant_url (mgrpDisallowedBroadcaster) outer level URLs
11	2048	ldap_cant_url inner level URLs
12	4096	ldap_auth_url (mgrpAllowedBroadcaster) outer level URLs
13	8192	ldap_auth_url inner level URLs
14	16384	ldap_moderator_url (mgrpModerator) URLs
15	32768	ldap_jettison_url (mgrpJettisonBroadcasters) URLs (new in 7.4-0.01)
16	65536	ldap_auth_mappingN generated outer URLs (new in 7.5-31)
17	131072	ldap_auth_mappingN generated inner URLs (new in 7.5-31)

Bit 0 is the least significant bit.

The default value of the `ldap_use_async` MTA option is 0, which means that asynchronous LDAP lookups are disabled by default in the MTA.

39.15.1.7 LDAP bind and connect MTA options: `ldap_username` (`ldap-dn`)

The `ldap_username` MTA option specifies the DN under which to bind for LDAP queries. This option, if set, overrides for MTA purposes the base-level `ugldapbinddn` option (in legacy configuration, the `local.ugldapbinddn` configutil parameter).

39.15.1.8 LDAP bind and connect MTA options: `max_urls` (integer)

The `max_urls` MTA option specifies the maximum number of URLs that may be active when reiteratively performing URL lookups; that is, this is the maximum degree of nesting of URL references. The default value as of MS 8.0 is 1024; previously, the default was 128.

39.15.2 Direct LDAP domain lookup MTA options

A number of MTA options relate to domain lookups in LDAP.

- `domain_failure` is a rewrite template to handle cases of LDAP errors during the MTA's fundamental domain LDAP lookup while rewriting.
- `domain_match_url` is available to support the *STRONGLY DISCOURAGED* use of so-called "vanity domains".
- `domain_uplevel` is important for domain aliasing, and in particular for implicit aliasing as in the case of supporting arbitrary subdomains of some upper domain name.
- `ldap_domain_known_attributes` may have performance implications with some LDAP servers. Proper setting is also important in cases of site MTA configuration with reliance upon use of additional, site-specific LDAP attributes from domain entries.
- Affecting interpretation of the results of domain lookups are `ldap_default_domain`, `ldap_host_alias_list`, and `ldap_local_host`.
- Controlling caching and timeouts of domain lookups are `domain_match_cache_timeout`, `domain_match_cache_size`, and `ldap_domain_timeout`.
- Schema options include `ldap_domain_filter_schemal`, `ldap_domain_filter_schema2` and `ldap_domain_root`.
- Options to rename which LDAP attributes are used/recognized in domain entries are discussed at [Direct LDAP attribute name MTA options](#); see especially the `ldap_domain_attr_*` MTA options.

39.15.2.1 Domain lookup failures (`domain_failure`)

The `domain_failure` MTA option specifies what rewriting to apply when a [rewrite rule \\$V LDAP lookup](#) encounters an LDAP error. The default is

```
reprocess-daemon$Mtcp_local$1M$1~-error$4000000?Temporary lookup failure
```

This means that if the [rewrite rule \\$V LDAP lookup](#) encounters a (presumably temporary) LDAP error such as the LDAP server failing to allow connections, or an LDAP query timing out without a response, then the MTA will either: (a) for attempted submissions via `tcp_local` (attempted submissions from the Internet) or from channels "internal" to the MTA (such as `reprocess` or `conversion` channels), reject the submission attempt with a temporary error (4yz error) so that the message remains where it was (on the external, Internet host, or on the internal channel) which can re-attempt submission later, or (b) for all other messages (in particular, messages submitted by "internal" user e-mail clients or "internal" hosts via channels such as `tcp_intranet`, `tcp_auth`, `tcp_submit`, *etc.*, accept the message but divert it to the `reprocess` channel, so that the user's message submission is completed, leaving it up to the MTA (via the `reprocess` channel) to later go to the work of re-trying the LDAP lookup. In case (a), for attempted submissions from the Internet, one typically does not want to accept a message that one cannot process at the moment (and for which, in particular, one cannot even validate that the domain is one one wants to accept). Accepting such messages merely clutters one's own queues which at best cannot be processed immediately, and which at worst may need to be rejected (once domain validation can be performed) as invalid domains. Instead, sites usually prefer to refuse such messages with a temporary rejection (so that the remote host knows to continue with additional submission attempts later). And in case (a), for attempted submissions from "internal" channels, the messages might as well stay in their

current "internal" channel and await additional lookup attempts from there. In contrast, in case (b), for attempted submissions from one's own, "internal" user e-mail clients, user e-mail clients cannot usually handle making an automated resubmission attempt "later". So it is friendlier to one's own users to go ahead and accept the messages, even if the messages will end up needing to be rejected later.

In more detail, the `$M` and `$1M` rewrite rule control sequences are used here to do a preliminary check whether the `tcp_local` or any "internal" channels are doing the rewriting (attempting to enqueue). Then the `$1~` [rewrite rule control sequence](#) is used to override that initial channel match success or failure with a forced success, while truncating the rewrite rule at this point if the initial channel match failed.

So the effect is that when the `tcp_local` or an "internal" channel is attempting to enqueue, a rewrite rule template of the form

```
$U%H@reprocess-daemon-error$4000000?Temporary lookup failure
```

is used. Because (normally) there is no channel with official host name `reprocess-daemon-error`, this rewrite rule template has the effect of forcing a channel match failure, so the `$?` [rewrite rule control sequence](#) comes into play, specifying that a "4.0.0 Temporary lookup failure" error be returned for the address. The MTA does not enqueue the message; the message remains where it was (and the remote host or internal channel can re-attempt submission later).

But if any channel other than `tcp_local` or an "internal" channel (in particular, if a channel such as `tcp_intranet`, `tcp_auth`, or `tcp_submit`) is attempting to enqueue, a rewrite rule template of the form

```
$U%H@reprocess-daemon
```

is used. Thus in this case (messages being submitted from channels such as `tcp_intranet`, `tcp_auth`, or `tcp_submit`), the rewrite rule succeeds in directing the message to the [reprocess channel](#); the MTA accepts the message and routes it to the `reprocess` channel for further processing (in particular, for further attempts to do the domain lookup in LDAP). The `reprocess` channel will then continue to try to process the message (in particular, reattempt the [rewrite rule \\$V LDAP lookup](#)) until either the LDAP lookup is completed and then the message can be processed further, or the message eventually times out (time out as controlled by the final value of the [notices channel option](#) applying to the [reprocess channel](#)) causing the message to be returned to the original sender. Accepting the message onto the MTA, for the MTA to later re-try the domain lookup typically makes sense in the case of messages submitted from one's own users.

39.15.2.2 Direct LDAP domain lookup URL (`domain_match_url`)

The `domain_match_url` option may be used to specify an additional, special lookup to perform when looking for domains during a [\\$V rewrite rule template LDAP lookup](#); (that is, a lookup in addition to the regular lookup for normal domains triggered by `$V`). This additional lookup will be performed only if the check for a normal, hosted domain fails. The main use of this option is to enable use of vanity domains, vanity domain being disabled by default. To enable the use of vanity domains, in Schema 1 mode set

```
domain_match_url=ldap:///§B?msgVanityDomain?sub?(msgVanityDomain=§D)
```

Note that use of vanity domains is **NOT RECOMMENDED!**

39.15.2.3 Subdomain handling in domain lookups (`domain_uplevel`)

The `domain_uplevel` MTA option affects how domain names are searched for and used in direct LDAP mode. The option takes a bit-encoded integer argument, where each bit controls a particular aspect of domain name searching/usage; see below. The default value is 0.

Table 39.10 `domain_uplevel` option bit values

Bit	Value	Usage
0	1	When set, domain map searches, such as the <code>§V</code> search in the typical configuration's <code>§*</code> rewrite rule , iterate with successive initial portions of the domain name stripped off until a match is found (or the domain name is exhausted). That is, with this bit set then a domain entry in the DC tree implicitly causes all subdomains of that domain to also "match" for purposes of domain lookups.
1	2	When set, searches on user addresses also look for the user address with the original domain name replaced by the domain name found during the domain map process (the "canonical" domain name). In particular, this can be useful either when bit 0 (value 1) is set (subdomains implicitly present due to the presence of a domain in the DC tree), or when aliased domains are in use.
2	4	Controls whether the domain name found during the domain map process (the "canonical" domain name) is compared with the configutil parameter <code>service.defaultdomain</code> (which can be overridden by the ldap_default_domain MTA option) when deciding whether an entry is in a hosted domain.
3	8	New in 6.1-0.01 . Check the "canonical" form the address, that is, the address with the domain part replaced by the canonical domain, against any mailEquivalentAddress attributes, and disable address reversal if any match occurs. This bit and bit 5 (value 32) are useful in preventing unwanted address rewriting when canonical domains are in use.
4	16	New in 6.3-0.15 . For address reversal purposes, do not reverse unless the address (original or possibly the original with domain replaced by the "canonical" domain) matches a mailAlternateAddress value. In particular, this allows mail domain aliases to effectively cause all users to implicitly have a mailEquivalentAddress value using the domain alias as the domain name.
5	32	New in 6.2-0.04 . Check the "canonical" form the address, that is, the address with the domain part replaced by the canonical domain, against the mail attribute, and disable address reversal if it matches.

Bit 0 is the least significant bit.

39.15.2.4 Direct LDAP attribute name MTA options: `ldap_attr_domain1_schema2` (LDAP attribute name)

The `ldap_attr_domain1_schema2` MTA option may be used to override, for MTA domain lookup purposes, the Domain Map library code's default use of the `sunPreferredDomain`

LDAP attribute as the name of the Schema 2 mode domain level LDAP attribute which specifies the domain name within the domain entry. If this MTA option is not set (the default), then the Domain Map's default of `sunPreferredDomain` is used, as normal.

39.15.2.5 Direct LDAP attribute name MTA options:

ldap_attr_domain2_schema2 (LDAP attribute name)

The `ldap_attr_domain2_schema2` MTA option may be used to override, for MTA domain lookup purposes, the Domain Map library code's default use of the `associatedDomain` LDAP attribute as the name of the Schema 2 mode domain level LDAP attribute which specifies any secondary domain names (aliases for the canonical domain name) within the domain entry. If this MTA option is not set (the default), then the Domain Map's default of `associatedDomain` is used, as normal.

39.15.2.6 Direct LDAP attribute name MTA options:

ldap_attr_domain_search_filter (LDAP attribute name)

The `ldap_attr_domain_search_filter` MTA option specifies the name of the LDAP attribute in the global configuration template area (see the [ldap_global_config_templates](#) MTA option) that is used to store the domain search filter template. For instance, one attribute that might be used for such a purpose (hence to which this option might be set) would be `inetDomainSearchFilter`.

39.15.2.7 Direct LDAP schema MTA options:

ldap_basedn_filter_schema1 (LDAP URL filter),

ldap_basedn_filter_schema2 (LDAP URL filter elements)

(New in JES MS 6.3-0.15.) The `ldap_basedn_filter_schema1` MTA option specifies the filter used to identify schema 1 domains when performing baseDN searches. The `ldap_basedn_filter_schema2` MTA option specifies additional filter elements used to identify schema 2 domains when performing baseDN searches. The default is that neither the `ldap_basedn_filter_schema1` MTA option nor `ldap_basedn_filter_schema2` MTA option is set. When these options are not set, then the values of `ldap_domain_filter_schema1` and `ldap_domain_filter_schema2`, respectively, are used if those options are set. But if none of these options are set, then the default for `ldap_basedn_filter_schema1` is `"(objectclass=inetDomain)"`, while the default for `ldap_basedn_filter_schema2` is the empty string.

39.15.2.8 Direct LDAP attribute interpretation options:

ldap_default_domain (string)

The `ldap_default_domain` MTA option specifies the domain name that is recognized and interpreted as the default domain *by the MTA*. In legacy configuration, this option, if set, overrides the configutil parameter `service.defaultdomain`. In Unified Configuration, this option, if set, overrides (for MTA purposes) the base level `defaultdomain` option -- but it would be more normal/preferable to only set `defaultdomain` so that all components of Messaging Server have a consistent choice of default domain.

In particular, in [delivery_options](#) substitutions, the canonical domain name for the domain in which a user is located is compared against the `ldap_default_domain`

value. If they match, then substitutions such as `$I` will not insert a domain name; that is, `ldap_default_domain` is the domain name for which (in typical configurations) usernames will be left "bare" when constructing a mailbox.

The `ldap_default_domain` also specifies the domain name that preferentially will be returned for [Sieve environment "domain" tests](#); if it is not set, then the MTA falls back to the `received_domain` if set, or otherwise the [official hostname](#) of the L channel.

39.15.2.9 Direct LDAP schema MTA options:

`ldap_domain_filter_schema1` (LDAP URL filter),

`ldap_domain_filter_schema2` (LDAP URL filter)

The default is that neither the `ldap_domain_filter_schema1` nor `ldap_domain_filter_schema2` option is set, neither at the MTA level nor at the [base](#) level. When these options are not set, then internal defaults in the domain map code are used, equivalent to:

```
ldap_domain_filter_schema1=(|(objectclass=inetDomain)(objectclass=inetdomainalias))
ldap_domain_filter_schema2=(objectclass=sunManagedOrganization)
```

39.15.2.10 `ldap_domain_known_attributes` Option

The `ldap_domain_known_attributes` MTA (and [base](#)) option controls whether the MTA's domain lookup LDAP queries request all domain attributes, *vs.* solely a hard-coded list of "known" domain attributes. The default of `-1` means to request all domain attributes; setting this option to `1` causes the MTA to request its hard-coded list of "known" domain attributes.

It has been claimed that the `ldap_domain_known_attributes` setting can have a performance impact in some LDAP server environments.

Note that if a site has configured the MTA to use any site-specific custom LDAP attributes, in addition to the normal set that the MTA is hard-coded to interpret, then it is important to use the default setting of `-1` so that LDAP domain queries will return those additional, custom LDAP attribute values.

39.15.2.11 Direct LDAP schema MTA options: `ldap_domain_root` (DN)

The `ldap_domain_root` MTA option specifies, for MTA purposes, the base DN for the domain portion of the DIT. This option, if set, overrides (for MTA purposes) the `base` option [dcroot](#) (or in legacy configuration, the `configutil` parameter `service.dcroot`). If neither the MTA option nor the `dcroot` base option (in legacy configuration, the `configutil` parameter) has been explicitly set, then the default is `o=internet`.

39.15.2.12 LDAP lookup cache MTA options:

`ldap_domain_timeout` (integer)

The `ldap_domain_timeout` option (available at both base and MTA levels) controls the retention time (in seconds) for entries in the domain map cache. The default is `-900`; as the value used is the absolute value of the `ldap_domain_timeout` setting, this corresponds to

15 minutes. If setting `ldap_domain_timeout` explicitly, set it to a positive value so that the MTA can detect that it has indeed been intentionally set.

39.15.2.13 `ldap_host_alias_list` Option Under `mta`

The `ldap_host_alias_list` MTA option specifies local host aliases for LDAP lookup result interpretation purposes. Specifying this option at MTA level overrides, for MTA purposes, the `base.ldap_host_alias_list` base option (`local.imta.hostnamealiases` configutil parameter in legacy configuration), thereby allowing the MTA to recognize a different set of such aliases than the Message Store recognizes. Neither the base nor MTA level option has a default value; if the base level option is set, it is used by the MTA unless the MTA level option has been explicitly set in which case the MTA uses the MTA level option's value.

The `ldap_host_alias_list` value takes a comma-separated list of up to 40 host aliases; each host alias may be at most 256 characters long; the total length of the entire list is limited to 1024 characters. (In IMS 5.2, the limits were smaller: at most 20 host aliases and each host alias at most 252 characters long.) New in Messaging Server 7.0.5.36, the MTA supports wild-carded values.

The `ldap_host_alias_list` value(s) are used by the MTA when deciding whether a domain's `mailRoutingHosts` value(s) or a user's `mailHost` value is "local" (this MTA itself). That is, once an LDAP lookup of a domain or user occurs, this option's value(s) affect the interpretation of the result of the LDAP lookup.

39.15.2.14 Direct LDAP attribute interpretation MTA options: `ldap_local_host` (string)

The `ldap_local_host` MTA option specifies the local hostname ([official host name](#) for the "I" channel) for LDAP lookup result interpretation purposes. If not set, it defaults to the value of the `hostname` Base option. If set, this option overrides the `hostname` Base option (in legacy configuration, the `local.hostname` configutil parameter). Normally `ldap_local_host` and `channel:1.official_host_name` should be set to match: the distinction is that the `ldap_local_host` value affects interpretation by the MTA of LDAP lookup results (as well as during initial installation/configuration controlling what hostname is generated for the "I" channel `official_host_name` and combined with standard channel prefixes to generate an appropriate `official_host_name` value for the other standard channels), whereas `channel:1.official_host_name` controls MTA address interpretation and processing in other contexts such as rewrite rules and SMTP server hostname defaults. But since initial installation/configuration normally sets the `channel:1.official_host_name` based on the `hostname` Base option value, they normally indeed "match".

Note that the `&/IMTA_HOST/` substitution value comes from `ldap_local_host` (which if not set explicitly, defaults to the value of the `hostname` Base option, or `local.hostname` in legacy configuration).

39.15.3 Direct LDAP usergroup lookup MTA options

There are a number of MTA options relating to [direct LDAP alias lookups](#) (including user lookups, group lookups, and mailing list lookups) and address reversal lookups.

- The `alias_urlN`, and `reverse_url` MTA options are the major options for defining the direct LDAP alias and reverse lookups, while the (not usually modified) `alias_magic` can

potentially affect the timing of when -- or even whether -- such direct LDAP lookups are performed.

- Additional options further refining/modifying the alias and reverse lookups include the [allow_unquoted_addrs_violate_rfc2798](#), [ldap_default_attr](#), [ldap_mail_aliases](#), [ldap_mail_reverses](#), [max_alias_levels](#), and [max_urls](#) MTA options.
- The [alias_entry_cache_*](#) and [reverse_address_cache_*](#) MTA options control the caching of alias and address reversal lookup results; for further discussion see the discussion of [LDAP lookup cache MTA options](#).
- The [defer_group_processing](#) MTA option affects the timing of the MTA's alias expansion of group and list entries, which has implications on operation of such aliases.
- The [ldap_user_root](#), [ldap_user_object_classes](#), and [ldap_group_object_classes](#) MTA options set the basic location in the DIT and objectClasses for the user/group entries in the DIT; see also [Direct LDAP attribute name MTA options](#) for the LDAP attributes expected/recognized in user and group entries.
- The [delivery_options](#), [group_dn_template](#), [ldap_uid_invalid_chars](#), and [aliasdetourhost_null_optin](#) MTA options affect the interpretation (and validity of values) of certain LDAP attributes found during alias lookups; see [Direct LDAP attribute interpretation MTA options](#) for further discussion.
- The [ldap_filter_reference](#), [ldap_hoh_filter](#), and [ldap_hoh_owner](#) MTA options relate to "head of household controls" or "parental controls" applicability to users or aliases; further discussion of "head of household" Sieve filters can be found in [Head of household Sieve filters](#).

39.15.3.1 Alias and address reversal MTA options: [alias_urlN](#) (URL)

The [alias_urlN](#) MTA options each specify a URL to query for alias lookups. If more than one of these options is set, then the URLs lookups specified are performed in the order specified by the [alias_magic](#) MTA option. The options are normally checked in numeric order, so [alias_url0](#), if specified, will be the first URL queried. Note that with the usual value of [alias_magic](#) (8764), the [alias_url3](#) option is not used.

Such alias lookups will be performed any time an envelope To address matches the local ("I") channel, or any channel marked with the [aliaslocal](#) channel option.

The URL (that is, the [alias_urlN](#) option value) must be specified using standard LDAP URL syntax as per [RFC 4516](#), with the following exception and special interpretations:

- The LDAP server and port are typically omitted, being instead specified via the [ldap_host](#) and [ldap_port](#) MTA options; (or if the MTA options are not explicitly set, the MTA will use more general options: in legacy configuration, the configutil parameters `local.ugldaphost` and `local.ugldappport`, or in Unified Configuration the [ugldaphost](#) and [ugldappport](#) base options). (Indeed, prior to Messaging Server 7.4-18.01, the host and port had to be omitted; but as of Messaging Server 7.4-18.01, specifying the host and port in the URL itself is supported.)
- The MTA makes a distinction between a completely omitted attributes field, which as per [RFC 2255](#) means to request the return of *all* attributes, and an attributes field consisting

of the asterisk character, *, which the MTA instead interprets as meaning to request the return of *all known-to-the-MTA attributes*, that is, all the attributes listed in [Table of MTA LDAP attribute name options](#). This distinction is available since for some directory setups, there may be a noticeable performance difference in LDAP directory response to one type of query (all attributes requested) *vs.* the other type of query (specific, though large, list of attributes requested).

- Various [substitution sequences](#) of the form "\$n" are available. A literal dollar sign must be represented by "\$\$".

The LDAP URL, before any substitutions, is limited to 256 characters in length (252 in iMS 5.2 and earlier); the substitutions may insert additional material and the length after such substitutions is limited to 1024 characters. Note that the substitution of known attributes when asterisk, *, is specified as the attribute to return, is not considered as part of the regular substitution; this substitution is performed at a later step and the length after this "known" attributes substitution is limited to 4096 characters.

`alias_url0`, if set, is normally looked up first (unless the [alias_magic](#) value has been changed). Next `alias_url1`, if set, *etc.* It is permissible to have "gaps" in the `alias_urlN` list; for instance, it is permissible to set `alias_url0` and `alias_url2` without setting `alias_url1`.

Since `alias_url0` is normally looked up first, in a typical direct LDAP configuration it is used to perform the "main" user/group lookup, with `alias_url1` optionally being used by those sites that need to do an additional, secondary lookup. In particular, `alias_url1` is typically used by those sites that need to support vanity domains, or it could be used by sites that do not support vanity domains but that need to support "old-style" catch-all addresses, (that is, sites that use the deprecated approach of defining a catch-all address by means of a user [mailAlternateAddress](#) attribute with a wildcard, rather than using the preferred approach of defining a domain level [mailDomainCatchallAddress](#) attribute).

As of 7.0.5, the default value for `alias_url0` is `ldap:/// $V? *?sub? $R`. Previously, there was no default and `alias_url0` had to be set explicitly. The other `alias_urlN` MTA options have no default value.

39.15.3.2 Direct LDAP usergroup lookup MTA options: `ldap_default_attr` (attribute name)

Some sites upgrading from previous software may be accustomed to using LDAP query URLs that do not specify an attribute to return (which for LDAP query URLs literally means to return *all* attributes) in places where all that they really wanted was the return of a single attribute. If the MTA sees an LDAP URL that does not specify which attribute(s) to return used in a place where the MTA knows that only a single attribute is desired, then the MTA will normally change the LDAP URL by forcibly inserting `mail` in the (omitted) *attributes* field of the LDAP URL.

The `ldap_default_attr` MTA option may be used to tell the MTA some other attribute to forcibly insert into the LDAP query URLs (some other attribute to request) in such cases where the *attributes* field was incorrectly omitted from the original LDAP query URL.

39.15.3.3 Direct LDAP usergroup lookup MTA options: `ldap_mail_aliases` (comma-separated list of attribute names)

The `ldap_mail_aliases` MTA option specifies in what attributes aliases are stored. Hence in particular, this option controls what attributes are used to construct the filter that a [\\$R LDAP substitution sequence](#) inserts, (note that the `$R` substitution sequence is typically used in the settings of both the [alias_url0](#) and [reverse_url](#) MTA options), as well as the attributes requested when doing an LDAP-based mailing list access check on an address. Up to ten, comma-separated attribute names may be specified.

The `ldap_mail_aliases` MTA option, if set, overrides the `local.imta.mailaliases` configutil parameter; if neither this option nor `local.imta.mailaliases` configutil parameter is explicitly set, then default values are used based upon the schema tag; (see the [ldap_schematag](#) MTA option). For a schema tag value of `ims50`, the default for the `ldap_mail_aliases` option is `"mail,mailAlternateAddress,mailEquivalentAddress"`. For a schema tag value of `nms41`, the default for this option is `"mail,mailAlternateAddress"`. For a schema tag value of `sims40`, the default for this option is `"mail,rfc822mailalias"`.

39.15.3.3.1 Use with mailaliases Under mta

List of comma-delineated LDAP attributes that override the default attributes. These attributes should be email addresses that can be routed.

39.15.3.4 Direct LDAP usergroup lookup MTA options: `ldap_mail_reverses` (comma-separated list of attribute names)

The `ldap_mail_reverses` MTA option specifies what attributes are used to build the filter referenced by the [\\$Q LDAP substitution sequence](#). (In iMS 5.2p2, the [reverse_url](#) MTA option, used for address reversal, typically made use of the `$Q` substitution sequence. However nowadays, [reverse_url](#) typically instead makes use of the `$R` substitution sequence with attribute list therefore determined via the [ldap_mail_aliases](#) MTA option. So nowadays `$Q`, and hence `ldap_mail_reverses`, are of lesser interest.)

The default, if this option is not explicitly set, depends upon the [schema tag](#). For a schema tag value of `ims50` or `nms41`, the default for the `ldap_mail_reverses` option is `"mail,mailAlternateAddress"`. For a schema tag value of `sims40`, the default for this option is `"mail,rfc822MailAlias"`.

See also the [ldap_mail_aliases](#) and [ldap_equivalence_addresses](#) MTA options.

Normally, `ldap_mail_reverses` should be set to include, in addition to the canonical `mail` attribute, all attributes set for `ldap_mail_aliases`, but should *not* include the attribute(s) set for [ldap_equivalence_addresses](#). In particular, if [ldap_mail_aliases](#) is changed to a non-default value, one would normally want to change `ldap_mail_reverses` in a corresponding fashion.

39.15.3.5 Direct LDAP schema MTA options: `ldap_user_root` (DN)

The `ldap_user_root` MTA option specifies the base DN for the user and group portion of the DIT for purposes of MTA LDAP URL lookups. (In particular, [\\$B substitutions in LDAP URL lookups](#) use this value.) This option, if set, overrides the base option [ugldapbasedn](#) (or in legacy configuration, the configutil parameter `local.ugldapbasedn`). If neither the MTA option nor the base option (in legacy configuration, the configutil parameter) has been explicitly set, then the default is `o=isp`.

39.15.3.6 Direct LDAP usergroup lookup MTA options: `reverse_url` (URL)

The `reverse_url` MTA option specifies the URL to query for address reversal and associated [side-effects](#). Standard LDAP URL syntax as per [RFC 2255](#) is used, except that the LDAP server and port may be omitted (in which case they are specified by the `ldap_host` and `ldap_port` MTA options -- or in legacy configuration, alternatively specified via the `configutil` parameters `local.ugldaphost` and `local.ugldapport`). Also, certain [substitution sequences](#) are available. The length, before substitution, is limited to 256 characters; (the limit was 252 characters in iMS 5.2 and earlier); the length resulting from the substitutions is limited to 1024 characters.

For typical MTA configurations, the usual value to which to set the `reverse_url` MTA option is `"ldap:/// $V? $N?sub? $R"`. Indeed, as of 7.0.5, this is the default; (previously there was no default and a value had to be set explicitly).

Note that the `$R` [substitution sequence](#), which is the filter for the LDAP query, uses the attributes named by the `ldap_mail_aliases` MTA option, (or if that option is not set in legacy configuration, the attributes named by the `local.imta.mailaliases` `configutil` parameter).

The reason that `reverse_url` normally is set to use a filter that searches on the canonical `mail` attribute, as well as the "subject to reversal" attributes such as `mailAlternateAddress`, is that `reverse_url` lookups actually do more than pure address reversal: `reverse_url` lookups also result in [setting other possibly desired information for messages](#), including for instance use of attributes named by the `ldap_personal_name`, `ldap_capture`, and `ldap_domain_attr_report_address` MTA options. Therefore, the canonical `mail` attribute is included in the search filter (included in `ldap_mail_aliases` which controls what is included in the filter referred to via the `$R` [substitution sequence](#)) so that lookups will succeed, and find desired information, even for those users whose addresses are *already* in canonical form.

39.15.4 Direct LDAP schema MTA options

A number of MTA options relate to overall schema choice, and layout of the LDAP Directory Information Tree. In addition to these general, semantically significant such options, there are also options to "rename" the LDAP attributes normally used in the schema, see the [Direct LDAP attribute name MTA options](#), as well as options controlling and altering the MTA's interpretation of LDAP attributes, see the [Direct LDAP attribute interpretation MTA options](#).

39.15.4.1 Direct LDAP attribute name MTA options:

`ldap_attr_domain_search_filter` (LDAP attribute name)

The `ldap_attr_domain_search_filter` MTA option specifies the name of the LDAP attribute in the global configuration template area (see the [ldap_global_config_templates](#) MTA option) that is used to store the domain search filter template. For instance, one attribute that might be used for such a purpose (hence to which this option might be set) would be `inetDomainSearchFilter`.

39.15.4.2 Direct LDAP schema MTA options:

`ldap_basedn_filter_schema1` (LDAP URL filter),

`ldap_basedn_filter_schema2` (LDAP URL filter elements)

(New in JES MS 6.3-0.15.) The `ldap_basedn_filter_schema1` MTA option specifies the filter used to identify schema 1 domains when performing baseDN searches. The `ldap_basedn_filter_schema2` MTA option specifies additional filter elements used to identify schema 2 domains when performing baseDN searches. The default is that neither the `ldap_basedn_filter_schema1` MTA option nor `ldap_basedn_filter_schema2` MTA option is set. When these options are not set, then the values of `ldap_domain_filter_schema1` and `ldap_domain_filter_schema2`, respectively, are used if those options are set. But if none of these options are set, then the default for `ldap_basedn_filter_schema1` is `"(objectclass=inetDomain)"`, while the default for `ldap_basedn_filter_schema2` is the empty string.

39.15.4.3 Direct LDAP schema MTA options: `ldap_domain_filter_schema1` (LDAP URL filter), `ldap_domain_filter_schema2` (LDAP URL filter)

The default is that neither the `ldap_domain_filter_schema1` nor `ldap_domain_filter_schema2` option is set, neither at the MTA level nor at the [base](#) level. When these options are not set, then internal defaults in the domain map code are used, equivalent to:

```
ldap_domain_filter_schema1=(|(objectclass=inetDomain)(objectclass=inetdomainalias))
ldap_domain_filter_schema2=(objectclass=sunManagedOrganization)
```

39.15.4.4 Direct LDAP schema MTA options: `ldap_domain_root` (DN)

The `ldap_domain_root` MTA option specifies, for MTA purposes, the base DN for the domain portion of the DIT. This option, if set, overrides (for MTA purposes) the base option `dcroot` (or in legacy configuration, the `configutil` parameter `service.dcroot`). If neither the MTA option nor the `dcroot` base option (in legacy configuration, the `configutil` parameter) has been explicitly set, then the default is `o=internet`.

39.15.4.5 Direct LDAP schema MTA options: `ldap_global_config_templates` (DN)

The `ldap_global_config_templates` MTA option specifies the base DN where global configuration templates can be found. It has no default. Note that this option should never be used under normal circumstances; if it is used to specify an unusual search scheme, it may result in domain inconsistencies and other problems.

39.15.4.6 Direct LDAP schema MTA options: `ldap_group_object_classes` (list of plus-separated list of objectclass names)

The `ldap_group_object_classes` MTA option specifies the object classes required to be present in a group entry. The default depends upon the schema tag (see the [ldap_schematag](#) MTA option). For a schema tag value of `ims50`, the default for the `ldap_group_object_classes` option is `inetLocalMailRecipient+inetmailgroup`.

For a schema tag value of `nms41`, the default for this option is `mailGroup`. For a schema tag value of `sims40`, the default for this option is `inetMailRouting+inetmailgroup`.

39.15.4.7 LDAP bind and connect options: `ldap_schemalevel` (1 or 2)

The `ldap_schemalevel` [base option](#) specifies the schema level in use. This option is also available at MTA level. Supported values are 1 or 2. If this option is not set, schema level 1 is assumed to be in use.

39.15.4.8 Direct LDAP schema MTA options: `ldap_schematag` (list of schema tags)

The `ldap_schematag` MTA option specifies the tag(s) for the schema in use. Valid values are `nms41`, `sims40`, or `ims50`, or a comma-separated list of these values. If specified, this option overrides the value of the `local.imta.schematag` configutil parameter in legacy configuration. If neither this option nor the `local.imta.schematag` configutil parameter is specified, then the default value assumed is `ims50`.

For purposes of authentication (*e.g.*, user IMAP or POP logins, or SMTP AUTH authentication), there is no corresponding option or configutil parameter setting an overall schema tag (hence automatically causing appropriate use of an alternate schema). Instead, see the [searchfilter](#) auth option (corresponding to the legacy configuration `sas1.default.ldap.searchfilter` configutil parameter), which has default value

```
(&(uid=%U)(objectclass=inetmailuser))
```

hence includes an implicit schema assumption. If using a non-default schema, that auth option (configutil parameter in legacy configuration) may need to be changed to cause user lookups (for authentication purposes) to look for objectclass(es) used in the other schema (*e.g.*, the NMS 4.1 schema).

39.15.4.9 Direct LDAP schema MTA options: `ldap_user_object_classes` (list of plus-separated list of objectclass names)

The `ldap_user_object_classes` MTA option specifies the object classes required to be present in a user entry.

The default depends upon the schema tag (see the [ldap_schematag](#) MTA option). For a schema tag value of `ims50`, the default for the `ldap_user_object_classes` option is `inetLocalMailRecipient+inetmailuser`. For a schema tag value of `nms41`, the default for this option is `mailRecipient+nsMessagingServerUser`. For a schema tag value of `sims40`, the default for this option is `inetMailRouting+inetmailuser`.

Note that the [\\$K LDAP URL substitution sequence](#) uses these object classes. For instance, when `ldap_user_object_classes=inetLocalMailRecipient+inetMailUser`, then `$K` results in substituting

```
(|(&(objectClass=inetLocalMailRecipient)(objectClass=inetMailUser)))
```

Or if `ldap_user_object_classes` is set to

```
inetLocalMailRecipient+inetMailUser,inetMailRouting+inetMailUser
```

then `$K` results in

```
( | (&(objectClass=inetLocalMailRecipient)(objectClass=inetMailUser))
  (&(objectClass=inetMailRouting)(objectClass=inetMailUser)))
```

For purposes of authentication (*e.g.*, user IMAP or POP logins, or SMTP AUTH authentication), the definition of valid user `objectClass(es)` is implicit in the setting of the `searchfilter` auth option (or in legacy configuration, the `sas1.default.ldap.searchfilter` configutil parameter), which has default value

```
(&(uid=%U)(objectclass=inetmailuser))
```

hence implicitly includes a schema assumption, and in particular an assumption about what are valid user `objectClass(es)`. If using a non-default schema, this configutil parameter may need to be changed to cause user lookups (for authentication purposes) to look for `objectclass(es)` used in the other schema (*e.g.*, the NMS 4.1 schema).

39.15.4.10 Direct LDAP schema MTA options: `ldap_user_root` (DN)

The `ldap_user_root` MTA option specifies the base DN for the user and group portion of the DIT for purposes of MTA LDAP URL lookups. (In particular, [\\$B substitutions in LDAP URL lookups](#) use this value.) This option, if set, overrides the base option `ugldapbasedn` (or in legacy configuration, the configutil parameter `local.ugldapbasedn`). If neither the MTA option nor the base option (in legacy configuration, the configutil parameter) has been explicitly set, then the default is `o=isp`.

39.15.5 Direct LDAP attribute interpretation MTA options

Some MTA options control or alter the MTA's interpretation of various LDAP attributes, or specify validity restrictions on LDAP attribute values.

39.15.5.1 User/group LDAP attribute validity and interpretation options: `aliasdetourhost_null_optin` (string)

(New in JES MS 6.2p4.) Normally, the simple presence of a `detour_optin` attribute (the attribute named by the `ldap_detourhost_optin` MTA option in a user entry, or new in 7.0.5, the attribute named by the `ldap_domain_attr_detourhostoptin` MTA option in a domain entry) suffices to cause message detour for messages coming in a channel marked `aliasoptindetourhost`; the value of the attribute is irrelevant. However, some directory maintenance and provisioning tools cannot easily delete or omit an attribute; instead, they always provide the attribute but assume that some "off" or "null" value for the attribute is available. The `aliasdetourhost_null_optin` option allows for better interaction with such directory tools. It specifies what value the `detour_optin` attribute (that attribute named by the `ldap_detourhost_optin` MTA option, or new in 7.0.5 the

`ldap_domain_attr_detourhostoptin` MTA option) must have in order to be ignored (for the MTA to act as if the attribute was not present at all in the user or domain entry). The default value for this option is the empty string, when means that by default a present but empty `detour optin` attribute is ignored.

Note that message detouring of the `aliasoptindetourhost` type is typically a detour to some third-party host or channel performing spam/virus filtering. Thus the value of `aliasdetourhost_null_optin` typically means what value to be considered as the "this user hasn't asked to do spam/virus filtering" value. But note that, for instance, use of such a null value in a user entry does not necessarily disable the detouring (the spam/virus filtering); a user who is not opted-in personally may still be detoured due to a domain level setting, or a channel level (`aliasdetourhost` channel option) setting.

39.15.5.2 Direct LDAP usergroup lookup MTA options: `allow_unquoted_addrs_violate_rfc2798` (0 or 1)

The default for the `allow_unquoted_addrs_violate_rfc2798` MTA option is 0. If set to 1, then when searching for an address match the MTA also includes in the search filter a version of the address with quotes stripped off the localpart (portion to the left of the @ character) of the address.

39.15.5.3 Direct LDAP attribute interpretation MTA options: `capture_format_default` (0 or 1)

(New in 7.4-18.01.) The `capture_format_default` MTA option controls whether `ldap_capture` based message "capture" defaults to generating regular, report style messages (the original message encapsulated in a form of DSN), *vs.* other possible forms, such as envelope "journal" style messages. The default is 0, meaning to generate report (DSN encapsulated) messages as the "capture" copies. A value of 1 means to generate pure, unadorned (no additional structure or header lines) messages as the "capture" copies. A value of 2 means to generate Microsoft® Exchange "journal" format messages as the "capture" copies. New in 8.0 are the values 4 and 5 which mean, respectively, to generate a DSN format message containing only the message header or an Microsoft Exchange "journal" format message containing only the message header, as the "capture" copy.

This option can be overridden on a per-capture-target-address basis by using the appropriate LDAP tag on the "capture" attribute (see `ldap_capture` and `ldap_domain_attr_capture`), where `;format-report` selects the DSN encapsulated format, and `;format-journal` selects the envelope "journal" format. New in 8.0, is support for the LDAP tags `;format-message`, `;format-report-header`, and `;format-journal-header`.

39.15.5.4 Direct LDAP attribute interpretation MTA options: `delivery_options` (list of strings)

The `delivery_options` MTA option controls the effect of possible values of the LDAP attribute named by the `ldap_delivery_option` MTA option, (by default, the `mailDeliveryOption` attribute). It takes a list of up to twenty strings. Each such string specifies the effect of a particular supported value for the `mailDeliveryOption` attribute. The syntax for an individual string (among the list of strings) is:

ModifierValue=Effect

where *Modifier* consists of one or more of the optional modifier letters listed in the following [table](#), where *Value* is a supported value for the `mailDeliveryOption` attribute, and where *Effect* describes the intended effect of the corresponding `mailDeliveryOption` value. Note that if neither `*` nor `&` is present, then the delivery option entry is taken to apply to both users and groups.

Table 39.11 Modifier letters for `delivery_options`

Modifier letter	Meaning
*	Entry applies to users
&	Entry applies to groups
\$	Expansion of this user or group should be deferred
@	Force to the reprocess channel
^	Check the vacation start and end time; only apply this entry if the current time is between the specified start and end times (or if no date/time restrictions are in effect).
%	(New in 7.0.5) Check the vacation start and end time; only apply this entry if the current time is outside those times (or if no date/time restrictions are in effect)
#	Entry is mailhost-independent; if all of a user or group's delivery options are mailhost-independent, then the MTA can act on the entry immediately rather than having to forward the message to the mailhost .
/	Force addresses produced by this delivery option to be sidelined in <code>.HELD</code> message files.
!	Use internal autoreply mode -- that is, generate a Sieve "vacation" scriptlet (rather than using the obsolete autoreply channel).

For instance, `mailbox` normally has the modifier `*` meaning that it applies (only) to users, whereas `members` normally has the modifier `&` meaning that it applies (only) to groups.

And while `autoreply` normally has the modifier `*` meaning that it applies (only) to users (and so normally mailing list and group entries cannot use `autoreply/vacation` functionality), if it is desired to allow mailing list and groups to generate their own `autoreply/vacation` messages, then removing the `*` modifier from the `autoreply` clause will allow this--- from the MTA point of view. (Note that the Sun schema as distributed normally does not expect/permit `mailAutoReply*` attributes to be set on mailing list or group entries. So for purposes of placating the Directory Server side, you will likely also need to either extend the schema, or disable schema checking.) As mentioned above, each *Value* should be a supported value for the `mailDeliveryOption` LDAP attribute (more precisely, the attribute named by the `ldap_delivery_option` MTA option). And each such supported value for `mailDeliveryOption` must have exactly one corresponding string in `delivery_options` describing its intended effect.

Each *Effect* specifies what happens to an original address that has a specified `mailDeliveryOption` value. For instance, a `mailDeliveryOption` value of `mailbox` (which is intended to mean delivery to the message store) is implemented by forcing the address local-part onto the [ims-ms channel](#) (the Message Store delivery channel); a `mailDeliveryOption` value of `native` is implemented by routing the address local-part to the native channel (the UNIX native mailbox delivery channel). The *Effect* definition may make use of [LDAP URL substitution sequences](#). In addition, an *Effect* of merely `*` means to

simply substitute back in the original address specified; an Effect of ** means to substitute the value of the mailForwardingAddress attribute (more precisely, the attribute named by the [ldap_forwarding_address](#) MTA option).

The current default for this option is (note that line breaks below are present merely for typographic reasons--the actual default value should be considered to appear all on one line):

```
*mailbox=$M%$\$2I$_+$2S@ims-ms-daemon, &members=*,
*native=$M@native-daemon, /hold=@hold-daemon:$1L+$2S@$D,
*unix=$M@native-daemon, &file=+$F@native-daemon,
&@members_offline=*, program=$M%$P@pipe-daemon,
#forward=**, *^!autoreply=$M+$D@bitbucket,
#*&nomail=$M+$D@bitbucket
```

This value first appeared in 8.0.1. The previous value, established in 7.0.5, differed in that it failed to preserve subaddresses in held messages:

```
*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+$D@bitbucket,
#*&nomail=$M+$D@bitbucket
```

Note the addition of the new-in-7.0.5 "nomail" clause. Setting a user to have the "nomail" delivery option causes the address to act as a valid recipient but silently delete all messages; this setting is useful for setting up an LDAP entry for a valid-but-unmonitored e-mail address. Formerly, prior to 7.0.5, the default had been:

```
*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
```

On a system doing LMTP delivery (via [LMTP client channels](#)), this option would normally be set to (on a JES MS 6.1 or later system):

```
#*mailbox=@$X.LMTP:$M$_+$2S@$\$2I, \
#&members=*, \
#*native=@$X.LMTPN:$M@native-daemon, \
#/hold=@hold-daemon:$A, \
#*unix=@$X.LMTPN:$M@native-daemon, \
#&file=@$X.LMTPN:+$F@native-daemon, \
#&@members_offline=*, \
#program=$M%$P@pipe-daemon, \
#forward=**, \
#^!autoreply=$M+$D@autoreply-daemon,
#*&nomail=$M+$D@bitbucket
```

where that assumes the use of rewrite rules along the lines of

```
.LMTP    $E$F$U%H.LMTP@lmtplib-daemon
.LMTP    $B$F$U%H@$H@lmtplib-daemon
.LMTPN   $E$F$U%H.LMTPN@lmtplib-native-daemon
.LMTPN   $B$F$U%H@$H@lmtplib-native-daemon
```

and two outbound `tcp_*` channels, (typically but not necessarily named `tcp_lmtpcs` and `tcp_lmtpcn`), corresponding respectively to the `lmtplib-daemon` and `lmtplib-native-daemon` official channel host names, and where both channels are marked with the `multigate` channel option.

Since up to twenty comma-separated strings may be specified for this option, note that up to twenty different possible delivery option values can be supported.

Note that the value clauses in the first two positions have special meaning as far as being the default delivery approaches for users and groups, respectively, which is why those first two value clauses are normally set to define `mailbox` and `members`. (That is, in the case of a user who has no `mailDeliveryOption` value specified in their LDAP entry, the first value clause of `delivery_options`---normally `mailbox`---will be assumed: a user with no `mailDeliveryOption` set gets messages delivered to their mailbox. Similarly, a group that has no `mailDeliveryOption` value specified will get the treatment specified by the second value clause of `delivery_options` -- normally `member`: a group with no `mailDeliveryOption` set gets messages delivered to the members of the group.) Thus if defining additional, site-specific mailbox delivery option values, be sure to add the custom values *later* in the list of option values.

Also note that when setting this option in the legacy configuration MTA option file `option.dat`, if using the backslash continuation line character to continue to additional lines, be aware of a potential confusion with "comment characters". Any line that begins in column one with one of the MTA option file's comment characters (`!`, `;`, `#`) will be interpreted as a comment *regardless* of whether the line above ended with a backslash. This issue can be worked around using the fact that the MTA ignores leading spaces after the comma separating individual strings within `delivery_options`; so you can use a definition such as

```
DELIVERY_OPTIONS=\
 *mailbox=$M%$ \$2I$_+$2S@ims-ms-daemon,&members=*,\
 *native=$M@native-daemon,hold=$M?$I@hold-daemon,\
 *unix=$M@native-daemon,&file=+$F@native-daemon,\
 &@members_offline=*,program=$M%$P@pipe-daemon,\
  #forward=**,^*!autoreply=$M+$D@bitbucket
```

where note the critical initial space on the line that does the `forward` value definition.

New in 7.0-0.04, there is some "sanity checking" on individual clauses within `delivery_options`, with an MM initialization error issued ("Invalid delivery option clause: `clause`") if such a check fails. Previously, certain sorts of problems in clauses would instead cause the clause to be silently ignored, with no warning or error.

In particular, prior to 7.0-0.04 the overall length limit for a clause was 81 characters, with at most 40 characters allowed left of the equals sign and at most 40 characters allowed right of

the equals sign. As of 7.0-0.04, the overall length limit for each clause is 256 characters (though exceeding this limit will merely cause silent truncation rather than an error), with at most 40 characters *not including leading modifier characters* to the left of the equals sign (that is, at most 40 characters in the actual "name" in the clause), and whatever remains of the 256 characters allowed on the right of the equals sign. Furthermore, as of 7.0-0.04, omission of an equals sign in a supposed clause will result in an MM initialization error.

39.15.5.5 Direct LDAP attribute interpretation MTA options: `group_dn_template` (URL template)

The `group_dn_template` MTA option affects the interpretation of the `uniqueMember` attribute (more specifically, the interpretation of the attribute named by the `ldap_group_dn` and as of 7.0.5, the `ldap_group_dn2`, MTA options). As of 7.4-18.01, the default is:

```
ldap:/// $A??sub?(mail=*)
```

The default (as of JES MS 6.3, but prior to 7.4-18.01) was:

```
ldap:/// $A??sub?mail=*
```

And prior to JES MS 6.3, the default had been:

```
ldap:/// $A?mail?sub?mail=*
```

Note that these earlier defaults prior to Messaging Server 7.4-18.01 were actually in error (violated LDAP search filter syntax), as they omitted the enclosing parentheses that ought to be present on the filter.

The change in the default for JES MS 6.3 (of no longer specifically requesting only the `mail` attribute to be returned) was to allow taking advantage, even in the case of an `mgrpAllowedBroadcaster` attribute set to a DN, of an enhancement (also implemented in JES MS 6.3) whereby list expansion in the context of the `mgrpAllowedBroadcaster` LDAP attribute now includes all the attributes used to store email addresses (normally `mail`, `mailAlternateAddress`, and `mailEquivalentAddress`), thereby allowing recognition of allowed broadcaster aliases. (Previously only `mail` attributes were returned, making it impossible to send to lists restricted to their own members using alternate addresses (aliases).)

39.15.5.6 Archive message format control: `journal_format` (bitmask)

The `journal_format` MTA option controls the `format` of Microsoft® Exchange journaling messages generated by the MTA. This is a bit-encoded option. Currently assigned bits are:

Table 39.12 `journal_format` MTA option bit values

Bit	Value	Description
0	1	If set, generate the basic 2007 journal format instead of the 2003 format.
1	2	If set, set the From:/To:/Subject: of the journal message to be the same as the message

		being journaled. Note that setting this may cause looping problems for setups that use header checks to determine what messages to archive.
2	4	If set, generate a X-MS-Exchange-Organization-Journal-Report: header field rather than a X-MS-Journal-Report: field.
3	8	If set, include expanded/forwarded address information in the report (if such information is available -- see the addrtypescan channel option). Note that bit 0 must also be set for this to work.

The default value for this option is 0. This option is intended to facilitate interoperating with Microsoft Exchange itself, in particular, so that MTA-generated journal messages can be imported into Microsoft Exchange.

39.15.5.7 Direct LDAP attribute interpretation options:

ldap_default_domain (string)

The `ldap_default_domain` MTA option specifies the domain name that is recognized and interpreted as the default domain *by the MTA*. In legacy configuration, this option, if set, overrides the configutil parameter `service.defaultdomain`. In Unified Configuration, this option, if set, overrides (for MTA purposes) the base level `defaultdomain` option -- but it would be more normal/preferable to only set `defaultdomain` so that all components of Messaging Server have a consistent choice of default domain.

In particular, in [delivery_options](#) substitutions, the canonical domain name for the domain in which a user is located is compared against the `ldap_default_domain` value. If they match, then substitutions such as `$I` will not insert a domain name; that is, `ldap_default_domain` is the domain name for which (in typical configurations) usernames will be left "bare" when constructing a mailbox.

The `ldap_default_domain` also specifies the domain name that preferentially will be returned for [Sieve environment "domain" tests](#); if it is not set, then the MTA falls back to the `received_domain` if set, or otherwise the `official_hostname` of the L channel.

39.15.5.8 Head of household LDAP attribute MTA options:

ldap_hoh_filter (LDAP attribute name), ldap_hoh_owner (LDAP attribute name)

The `ldap_hoh_filter` and `ldap_hoh_owner` MTA options specify the names of the LDAP attributes used to store the critical Head of Household data in users who are themselves a Head of Household. These options default to, respectively, `mailSieveRuleSource` and `mail`. That is, `ldap_hoh_filter` specifies the name of the LDAP attribute in which a Head of Household user stores the Sieve filter used for Head of Household purposes (which may or may not be a different Sieve filter than the user's own, personal Sieve filter; the default `mailSieveRuleSource` value causes the Head of Household Sieve filter to be the same as the user's personal Sieve filter, but sites that wish a distinction may set `ldap_hoh_filter` to point to a different, site-specific LDAP attribute). Since proper evaluation of (and especially error reporting regarding) a Sieve filter requires an "owner" e-mail address associated with

that Sieve filter, the `ldap_hoh_owner` MTA option specifies what LDAP attribute in the Head of Household user's entry will be the address associated with the Sieve; again, the default value of `mail` means that the Head of Household user's own, personal e-mail address will be used, but sites that wish a distinction may set `ldap_hoh_owner` to some different, site-specific LDAP attribute.

These MTA options specify the names of the LDAP attributes to return when a user entry has parental controls/head of household controls set on it (see the `ldap_parental_controls` MTA option) so that a lookup of the user's "parent" (see the `ldap_filter_reference` MTA option) is performed: in the "parent" entry, the attributes specified by `ldap_hoh_filter` and `ldap_hoh_owner` are found and their values returned.

39.15.5.9 `ldap_host_alias_list` Option Under `mta`

The `ldap_host_alias_list` MTA option specifies local host aliases for LDAP lookup result interpretation purposes. Specifying this option at MTA level overrides, for MTA purposes, the `base.ldap_host_alias_list` base option (`local.imta.hostnamealiases` configutil parameter in legacy configuration), thereby allowing the MTA to recognize a different set of such aliases than the Message Store recognizes. Neither the base nor MTA level option has a default value; if the base level option is set, it is used by the MTA unless the MTA level option has been explicitly set in which case the MTA uses the MTA level option's value.

The `ldap_host_alias_list` value takes a comma-separated list of up to 40 host aliases; each host alias may be at most 256 characters long; the total length of the entire list is limited to 1024 characters. (In IMS 5.2, the limits were smaller: at most 20 host aliases and each host alias at most 252 characters long.) New in Messaging Server 7.0.5.36, the MTA supports wild-carded values.

The `ldap_host_alias_list` value(s) are used by the MTA when deciding whether a domain's `mailRoutingHosts` value(s) or a user's `mailHost` value is "local" (this MTA itself). That is, once an LDAP lookup of a domain or user occurs, this option's value(s) affect the interpretation of the result of the LDAP lookup.

39.15.5.10 Direct LDAP attribute interpretation MTA options: `ldap_local_host` (string)

The `ldap_local_host` MTA option specifies the local hostname (`official host name` for the "I" channel) for LDAP lookup result interpretation purposes. If not set, it defaults to the value of the `hostname` Base option. If set, this option overrides the `hostname` Base option (in legacy configuration, the `local.hostname` configutil parameter). Normally `ldap_local_host` and `channel:l.official_host_name` should be set to match: the distinction is that the `ldap_local_host` value affects interpretation by the MTA of LDAP lookup results (as well as during initial installation/configuration controlling what hostname is generated for the "I" channel `official_host_name` and combined with standard channel prefixes to generate an appropriate `official_host_name` value for the other standard channels), whereas `channel:l.official_host_name` controls MTA address interpretation and processing in other contexts such as rewrite rules and SMTP server hostname defaults. But since initial installation/configuration normally sets the `channel:l.official_host_name` based on the `hostname` Base option value, they normally indeed "match".

Note that the `&/IMTA_HOST/` substitution value comes from `ldap_local_host` (which if not set explicitly, defaults to the value of the `hostname` Base option, or `local.hostname` in legacy configuration).

39.15.5.11 Direct LDAP attribute interpretation MTA options: `ldap_uid_invalid_chars` (list of integers)

This option specifies the ASCII positions of those characters which are not allowed to appear in a `uid`. (The MTA unconditionally disallows all characters below position 32, so this option specifies the list of additional characters to disallow.) The default is

```
32,33,34,35,36,37,38,40,41,42,43,44,47,58,49,60,61,62,63,65,91,92,93,96,123,125,126
```

which corresponds to the characters

```
$ ~=#*+%!@,{ } ( ) \ < > ; : " ` [ ] & ?
```

(space character and dollar character have been swapped for readability). Furthermore, note that the Message Store code further enforces a restriction that the leading character of the `uid` may not be a hyphen, `-`. (This is to avoid ambiguity with IMAP ACL syntax.) Prior to Messaging Server 7.0.5, The MTA does not enforce this restriction, however.

Note that the `uid` (synonym for `userID`) LDAP attribute was defined in [RFC 1274](#), The COSINE and Internet X.500 Schema, as a `caseIgnoreString` of length at most 256 characters. As of Messaging Server 7.0-0.04, the MTA checks that the `uid` value (more precisely, the value of the attribute named by the `ldap_uid` MTA option) is no more 128 octets, and a longer value will result in the user entry being considered invalid. (This check is performed because various lower layer libraries have hard buffer limits that preclude longer `uids`.) In general, because with Messaging Server the `uid` is used not only for logging in (a "computer system login name" is how [RFC 1274](#) discussed `userid`), but also, in hashed form, to specify part of the file path for where user messages are stored, then Messaging Server needs additional restrictions on the `uid` so that the file path constructed using the `uid` is good and safe.

39.15.5.12 Spamfilter MTA options: `optin_user_carryover` (bitmask)

New in JES MS 6.2. The `optin_user_carryover` MTA option controls whether user spam/virus filter "opt in" requests will "carry over" when doing forwarding. That is, if the original recipient has opted-in but has then forwarded their e-mail to some other recipient, does that other recipient get the "opt in" effect?

Bit 0 (value 1): setting this bit means that "opt in" effect is disabled for all forwarded-to address(es). Bit 1 (value 2) controls the behavior for [domain "opt in"](#); setting the bit disables the "opt in" effect for the forwarded-to address(es). Bit 2 (value 4) means that [user "opt in"](#) overrides any previous user/domain "opt in" setting. Bit 3 (value 8) controls the behavior for aliases (typically lists) marked with the [alias_optin](#) alias option or [named parameter \[OPTIN\]](#); setting the bit disables the "opt in" effect for the forwarded-to address(es). The default is 0. Note that this option applies globally to *all* spam/virus filter packages; it does *not* come in numbered variants to apply only to one spam/virus filter package or another.

39.15.5.13 Direct LDAP attribute interpretation MTA options: `process_substitutions` (bit-encoded integer)

New in JES MS 6.3. The `process_substitutions` MTA option controls whether to process substitution sequences in the URL values of various LDAP attributes. See [Table of LDAP URL substitution sequences](#) for a list of substitution sequences (though only some such substitution sequences make sense and are available in the contexts discussed below). The default is 0, meaning that all such substitutions are disabled by default.

Table 39.13 `process_substitutions` MTA option bits

Bit	Value	Usage
0	1	If set, enables substitutions in <code>mgrpDisallowedBroadcaster</code> (ldap_cant_url)
1	2	If set, enables substitutions in <code>mgrpAllowedBroadcaster</code> (ldap_auth_url)
2	4	If set, enables substitutions in <code>mgrpModerator</code> (ldap_moderator_url)
3	8	If set, enables substitutions in <code>mgrpDeliverTo</code> (ldap_group_url1)
4	16	If set, enables substitutions in <code>memberURL</code> (ldap_group_url2)
5	32	(New in Messaging Server 7.0) If set, enables subaddress <code>\$\$</code> substitution in <code>mgrpErrorsTo</code> (ldap_errors_to)
6	64	(New in Messaging Server 7.0u3) If set, enables substitutions in <code>mgrpJettisonBroadcasters</code> (ldap_jettison_url)

Bit 0 is the least significant bit.

Note that the information source for substitution values varies depending on whether the attribute in question is used for authorization checks, or for actual list expansion. For authorization attributes, the whole address (`$$`), domain (`$$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 (`$$`) is derived from the current envelope recipient address.

In particular, the ability to access subaddress information in list expansion URLs makes it possible and convenient to define a "meta-group"; that is, a single group entry that in effect creates an entire collection of different groups. For example, a group with attributes including:

```
mail: group@domain.com
mgrpDeliverTo: ldap:///o=usergroup?mail?sub?(department=$$)
```

would make it possible to send mail to every member of a given department with an address of the form

```
group+department@domain.com
```

Note that creation and use of such a "meta-group" does not require the use of subaddresses (though subaddresses are often a convenient syntax for such a purpose). Other mechanisms, such as other forms of "special" addresses transformed via a [FORWARD mapping table](#) or [ldap_url_result_mapping](#) attribute's value mapping table, could be used instead to

provide "meta-group" functionality. Note that `process_substitutions` effects, if any, occur *after* the `ldap_url_result_mapping` table, if any, has been applied.

39.15.5.14 Direct LDAP attribute interpretation MTA options: `route_to_routing_host` (0 or 1)

When a domain entry includes the attribute named by the `ldap_domain_attr_routing_hosts` MTA option, (by default, the `mailRoutingHosts` attribute), this attribute's values are compared against the MTA's own local host name and host aliases (`ldap_local_host` and `ldap_host_alias_list` values) to determine whether the domain is "owned" (or "local") to this particular MTA. "Local" addresses are of course further processed by the MTA.

The `route_to_routing_host` MTA option controls what the MTA does with addresses determined in this way to be non-local; that is, what the MTA does with addresses which, while having a local (known) domain in the directory, are marked as having some other authoritative mail host. When the `route_to_routing_host` MTA option is set to 0 (the default), such addresses are handled as specified by whatever rewrite rules apply to the address. (This was the only behavior available with iMS 5.2.) When this option is instead set to 1, then such addresses are instead routed to the first host listed in the `mailRoutingHosts` attribute.

39.15.5.15 Sieve filter MTA options: `sieve_user_carryover` (0 or 1)

New in JES MS 6.0-0.01. The default is 0. If set to 1, `user Sieve filters` don't "carry over" when doing `mailDeliveryOption: forward`. This option is only relevant for direct LDAP forwarding (forwarding via `mailDeliveryOption` and `mailForwardingAddress`); it does not have any effect on other forms of forwarding.

39.15.5.16 Handling of multiple spare LDAP attributes: `spare_N_separator` (bit-encoded integer)

The `spare_N_separator` MTA options, where N is between 1 and 18, control how multiple user entry LDAP attributes that end up being mapping into a single spare LDAP attribute slot are handled. These options accept a nonnegative integer value. The lower 8 bits of this value are interpreted as follows:

Table 39.14 `spare_N_separator` MTA option values

Value	Meaning
3	Multiple attribute values are not allowed - the user entry is considered invalid and ignored if multiples are present.
2	Use language tag information to decide which of the multiple attributes to use.
1	Multiple attributes are not allowed - the user entry is considered invalid if multiples are present.
0	Pick one of the values at random and use it.
32-255	Concatenate multiple attribute values together, using the character corresponding to the <code>spare_N_separator</code> value as the separator.

The remaining bits of the option are used as bit flags. Currently only bit 8 (value 256) is defined: If set, it causes the attribute name and an equals sign to be prepended to the stored value.

These options first became available in the Messaging Server 7.2-7.02 release. The default values for `spare_N_separator` are chosen to remain backwards compatible with spare attribute behavior in earlier releases, which was hardwired for a given spare slot. `spare_1_separator` and `spare_2_separator` default to 1, `spare_3_separator` defaults to 0, and `spare_4_separator`, `spare_5_separator`, and `spare_6_separator` default to 2. All of the remaining options up to `spare_18_separator` default to 0.

39.15.5.17 Autoresponse periodicity MTA options: `vacation_minimum_timeout` (integer)

(New in 7.0.5.) The `vacation_minimum_timeout` MTA option establishes a minimum value, in seconds, for the Sieve "`vacation`", "`:days`", "`:hours`", and "`:seconds`" parameters. ("`:days`" and "`:hours`" values are converted into seconds for the comparison.) Values lower than the minimum are silently adjusted up to the minimum; no error occurs. The default value for `vacation_minimum_timeout` is 0.

Since the value of the `mailAutoReplyTimeOut` LDAP attribute (or more precisely the value of whatever LDAP attribute is named by the [ldap_autoreply_timeout](#) MTA option) is converted into such a Sieve "`vacation`" parameter, the `vacation_minimum_timeout` MTA option value can affect the interpretation of any `mailAutoReplyTimeOut` values also.

39.15.5.18 Autoresponse periodicity MTA options: `vacation_maximum_timeout` (integer)

(New in 7.0.5.) The `vacation_maximum_timeout` MTA option establishes a maximum value, in seconds, for the Sieve "`vacation`", "`:days`", "`:hours`", and "`:seconds`" parameters. ("`:days`" and "`:hours`" values are converted into seconds for the comparison.) Values higher than the maximum are silently adjusted down to the maximum; no error occurs. The default value for `vacation_maximum_timeout` is the maximum allowed integer, $2^{31}-1$.

Since the value of the `mailAutoReplyTimeOut` LDAP attribute (or more precisely the value of whatever LDAP attribute is named by the [ldap_autoreply_timeout](#) MTA option) is converted into such a Sieve "`vacation`" parameter, the `vacation_maximum_timeout` MTA option value can affect the interpretation of any `mailAutoReplyTimeOut` values also.

39.15.6 Direct LDAP attribute name MTA options

By default, the MTA assumes a particular sort of LDAP schema; that is, the MTA assumes that certain named attributes (with certain sorts of meanings) are available and used in the LDAP directory to store the user and domain information. However, the exact attribute names that the MTA looks for (recognizes) are configurable via the various `ldap_*`, `ldap_attr_domain*`, and `ldap_domain_attr_*` MTA options, listed below. Thus a different (though semantically compatible) schema may be used by setting the `ldap_*`, `ldap_attr_domain*`, and `ldap_domain_attr_*` MTA options to tell the MTA what named attributes to use (recognize).

Note that many of the attributes used (and hence the attribute name which the MTA by default expects to see used) are standardized; see for instance [RFC 2798 \(Definition of the inetOrgPerson LDAP Object Class\)](#). Other attributes are specific to the Sun schema; see the *Sun Schema Reference Guide*.

Note that prior to JES MS 6.3-0.15, each LDAP attribute could be used for only one (from the MTA's point of view) purpose. In particular, prior to JES MS 6.3-0.15, the MTA would not permit setting two of its LDAP attribute name options to the same underlying LDAP attribute. If a site wanted to use the "same" LDAP attribute for multiple purposes in the MTA, that previously would have to be achieved by creating a second LDAP attribute (named differently), and having its value be duplicated in LDAP. New in JES MS 6.3-0.15, this restriction has been relaxed, so that two MTA purposes (options) can use the same underlying LDAP attribute; for instance, one can now set, say, `ldap_optin1` and `ldap_optin2` to both point to (use/name) the same underlying LDAP attribute, *e.g.*, `mailAntiUBEService`.

Note that throughout this discussion and other MTA discussions, for convenience often LDAP attributes will be referred to merely by name. But in general, any such MTA reference to a specific attribute name really ought to be a reference to the attribute named by the corresponding MTA option. For instance, any use by the MTA of the `mailConversionTag` attribute is really a use of the attribute named by the `ldap_conversion_tag` MTA option.

However, the general authentication libraries in Messaging Server (sometimes referred to as SASL libraries, or HULA) used for authentication (both by the MTA when performing SMTP AUTH authentication, or by the Message Store when performing login to a user mailbox) do not permit the same degree of "renaming" of attributes. As the authentication infrastructure uses LDAP simple bind for traditional password authentication, if the LDAP directory itself is configured to look at an attribute other than the usual `userPassword` for LDAP simple bind, that should just work. However, in order to support CRAM-MD5/APOP, then the `userPassword` attribute must be used and it must contain the clear-text password. The authentication infrastructure also has hard dependence on various user attributes including `uid`, `inetUserStatus`, `mailUserStatus`, and `mailAllowedServiceAccess` (among others). (Note that the MMP and its proxy servers can be configured to use a different LDAP attribute in place of `mailAllowedServiceAccess` via their `tcpaccessattr` option; the IMAP, POP, and MSHTTP servers, however, always use `mailAllowedServiceAccess`.)

And of particular relevance when configuring and considering MTA operation, another attribute which is not renameable (prior to the 8.0 release) via an MTA option is the `mailSMTPSubmitChannel` user attribute. (This is because the MTA itself makes no explicit use of this attribute. Instead, authentication library code explicitly fetches the `mailSMTPSubmitChannel` attribute's value, and then uses that value to tell the MTA what source channel to set.) But as of 8.0, some [renaming/specification of the attributes returned with successful authentication](#) is possible; in particular, see the `ldap_auth_attr_submit_channel` MTA option which specifies the name of the LDAP attribute whose value the authentication library should fetch (in place of the default `mailSMTPSubmitChannel` attribute's value). Also new in 8.0, the authentication library may be directed to fetch back values of LDAP attributes other than the default `mail` and `mailHost` via the `ldap_auth_attr_sender` and `ldap_auth_attr_mail_host` MTA options, respectively. See [Direct LDAP attributes returned upon authentication MTA options](#).

The schema sets restrictions (via an [ACI](#)) on which attributes even in his or her "own" entry an end user is allowed to modify. Reassigning the MTA's interpretation of LDAP attributes via MTA options does not, itself, affect such LDAP schema restrictions; so when reassigning end-user-modifiable LDAP attributes, be sure to also update your schema ACIs correspondingly.

Technical note: In the [table below](#), the user/group attributes are listed in roughly the order in which they are processed by the MTA (though there have been some changes in various versions, and there are some subtleties not captured in the order shown). While this order does not matter for most purposes, on occasion it can be helpful to consider this order as an aid to understanding certain interactions and precedence between attributes.

Table 39.15 MTA LDAP attribute name options

Option	Default attribute name(s)	Valid	Meaning and notes
Per-user/group attributes			
<code>ldap_objectclass</code>	<code>objectClass</code>	UGD	
<code>ldap_user_status</code>	<code>inetUserStatus</code>	U	<p>Prior to Messaging Server 7.0, the supported values were a strict subset of the supported <code>mailUserStatus</code> values, and in particular the only supported values were <code>active</code>, <code>inactive</code>, or <code>deleted</code>. As of Messaging Server 7.0, for the convenience of sites that may wish to "switch" the use (in effect switch the priority order in which checking occurs) of <code>inetUserStatus</code> and <code>mailUserStatus</code>, the full set of values supported for <code>mailUserStatus</code> are also supported for <code>inetUserStatus</code>. This is not intended to encourage general, direct use of such additional values for <code>inetUserStatus</code>, but rather, as mentioned, is intended so that the priority (order of checking) of these two status settings for users can be reordered by setting them "switched":</p> <p><code>ldap_user_status=mailUserStatus</code> <code>ldap_user_mail_status=inetUserStatus</code></p>
<code>ldap_user_mail_status</code>	<code>mailUserStatus</code>	U	<p>Valid values are <code>active</code>, <code>inactive</code>, <code>disabled</code>, <code>deleted</code>, <code>overquota</code>, <code>hold</code>, and <code>new</code> in JES MS 6.0 removed (which from the MTA point of view is equivalent to <code>deleted</code>; it exists as a distinct status for the benefit of the <code>commcli</code> user purging operation); and <code>new</code> in JES MS 6.3 <code>defer</code> and <code>defer-submit</code> (which, respectively, mean to accept all messages to the user but defer them to the reprocess channel for later delivery (re)attempts; or in the case of <code>defer-submit</code> means to accept and defer to the reprocess channel those messages coming in a <code>submit</code> channel while giving <code>inactive</code> behavior, hence normally temporary errors, for attempted submissions on any other channels); and <code>new</code> in 7.3-11.01? the value <code>deliver</code> which the MTA will treat as <code>active</code> for purposes of message delivery but which other components will treat as <code>inactive</code> (giving the effect that messages can be delivered, but the user can not login); any other value is treated as <code>inactive</code>.</p>
<code>ldap_group_status</code>		G	<p>Prior to Messaging Server 7.0, the supported values were a strict subset of the values supported for <code>inetMailGroupStatus</code>; in particular, the supported values were <code>active</code>, <code>inactive</code>, and <code>deleted</code>. New in Messaging Server 7.0, all the values supported for <code>inetMailGroupStatus</code> are supported for this attribute as well, for the convenience of sites that wish to "switch" the priority (order) in which they are checked by "switching" which attributes the MTA options <code>ldap_group_status</code> and <code>ldap_group_mail_status</code> point to.</p>
<code>ldap_group_mail_status</code>	<code>inetMailGroupStatus</code>	G	<p>Supported values are <code>active</code>, <code>deleted</code>, <code>removed</code>, <code>disabled</code>, <code>hold</code>, <code>inactive</code>, and <code>new</code> in Messaging Server 7.0 <code>defer</code> and <code>defer-submit</code>.</p>
<code>ldap_uid</code>	<code>uid</code>	UG	<p>As of JES MS 6.2, the MTA checks that there is only one such attribute; as of JES MS 6.3, the MTA also checks that there is only one value set for the one attribute. As of 7.0, the MTA checks that the UID value is no more than 128 octets; a longer value will result in the user entry being considered invalid. (This check is performed because various lower layer libraries have hard buffer limits that preclude longer UIDs.)</p>
<code>ldap_mlsrange</code>		UG	<p>(New in Messaging Server 7.0?) RESTRICTED</p>
<code>ldap_capture</code>		UG	<p>Specify an attribute used to trigger automatic capturing of user e-mail messages. The value of the attribute should be the address to which the "captured" messages should be sent. Typically, this attribute is set up so that it is not even visible, let alone modifiable, by the users themselves. When a user has this attribute specified on their entry, both messages sent to them, as well as from them, will also have a "capture" copy (an encapsulated copy with an entirely new message envelope) sent to the specified address. New in 7.0.5, the <code>capture_format_default</code> MTA option controls whether message copies generated due to use of the LDAP attribute named by <code>ldap_capture</code> default to being in DSN encapsulated format, or to being in envelope "journal" format. Also new in 7.4-18.01, values of the LDAP attribute may be tagged to explicitly specify the format on a per-target-address basis: the tag <code>;format-report</code> selects the usual DSN encapsulated format, whereas the tag <code>;format-journal</code> selects the envelope "journal" format.</p>
<code>ldap_recipientlimit</code>		UG	<p>Specify an attribute used to store a sending-user-specific maximum number of envelope recipients (additional recipients are rejected), analogous to the <code>recipientlimit</code> channel option. New behavior in JES MS 6.3 is that a per-user setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular user can be allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.</p>
<code>ldap_recipientcutoff</code>		UG	<p>Specify an attribute used to store a sending-user-specific maximum number of envelope recipients (messages with more recipients are rejected entirely), analogous to the <code>recipientcutoff</code> channel keyword. New behavior in JES MS 6.3 is that a per-user setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular</p>

Direct LDAP attribute name MTA options

			user can be allowed to send messages to a large number of recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.
ldap_sourceblocklimit		UG	Specify an attribute used to store a sending-user-specific maximum message size, analogous to the sourceblocklimit channel option. New behavior in JES MS 6.3 is that a per-user setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular user can be allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.
ldap_source_channel		UG	(New in 6.3) Specify a source channel to which to "switch" (if userswitchchannel is set on the current source channel)
ldap_source_optinn		UG	(New in 6.3) Sending user analogue of ldap_optinn option
ldap_preferred_language	<code>preferredLanguage+</code>	UG	The MTA's typical NOTIFICATION_LANGUAGE mapping table and DISPOSITION_LANGUAGE mapping table checks the value of this attribute when deciding in what language to send back notification messages. Also, as of JES MS 6.3, the MTA has the ability to chose between multiple LDAP 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. In JES MS 6.3, the only attributes receiving this treatment are ldap_autoreply_subject (normally <code>mailAutoReplySubject</code>), ldap_autoreply_text (normally <code>mailAutoReplyText</code>), ldap_autoreply_text_internal (normally <code>mailAutoReplyTextInternal</code>), ldap_spare_4 and ldap_spare_5 . As of Messaging Server 7.0-3.01, the attribute named by (new in that version) ldap_spare_6 also received such treatment; as of Messaging Server 7.2-7.01, any of the ldap_spare_N named attributes may optionally, depending upon the setting of the corresponding spare_N_separator MTA option, receive <code>preferredLanguage</code> treatment; and as of Messaging Server 7.3-11.01, the attribute named by ldap_add_tag also receives <code>preferredLanguage</code> treatment.
ldap_preferred_country		UG	(New in MS 6.3-0.15)
ldap_nosolicit		UG	New in 6.2-0.01. Specifies solicitation strings used by the SMTP NO-SOLICITING extension that this user doesn't want to receive.
ldap_routing_address	<code>mailRoutingAddress</code>	UG	Used to specify an address to which to route, overriding (as of JES MS 6.0) the usual <code>mailHost</code> check and <code>mailDeliveryOption</code> interpretation.
ldap_delivery_option	<code>mailDeliveryOption+</code>	UG	See the MTA option delivery_options for a discussion of the interpretation of possible values for this attribute.
ldap_personal_name		UG	Specify an attribute used to store a user's personal name. If this option is set, then the value of the specified attribute (if present in a user entry) will be inserted by the MTA as a personal name wherever the user's address appears in message headers (overriding any originally present personal name for the user that might have been present), including when generating vacation messages on behalf of the user. Note that (as of 6.2p3 for normal messages, or as of 6.2p6 for generated messages such as vacation messages) the MTA will quote the value obtained from LDAP, if required according to the quoting rules for personal names (technically "phrases") given in RFC 5322 .
ldap_source_conversion_tag		UG	New in JES MS 6.2. Specify an attribute whose value will be applied as a conversion tag for messages coming from this user.
ldap_primary_address	<code>mail</code>	UG	
ldap_alias_addresses	<i>varies with the schema tag; <code>mailAlternateAddress</code> for <code>ims50</code> or <code>nms41</code>; <code>rfc822mailalias</code> for <code>sims40</code></i>	UG	Attributes whose values (addresses) are accepted as equivalent to (aliases for) the canonical <code>mail</code> address on incoming messages; see also the ldap_mail_reverses MTA option which controls just which attributes (addresses) are normally converted to the canonical <code>mail</code> address during reverse_url application via the \$Q substitution sequence.
ldap_equivalence_addresses	<code>mailEquivalentAddress</code>	UG	Addresses accepted as equivalent to the canonical <code>mail</code> address for incoming messages; such equivalent addresses are also allowed to appear on outgoing messages (are not converted during reverse_url application). Multiple, comma-separated attribute names are permitted. Note that when setting this option to a non-default value, it is also usually appropriate/necessary to modify the ldap_mail_aliases MTA option correspondingly (to include the attribute(s) named by ldap_equivalence_addresses).
ldap_optin		UG	An alias for ldap_optin1 . The presence in a user entry of the attribute named by this option normally (but see the spamfilter_null_optin MTA option) causes messages addressed to this user to be "opted-in" for virus/spam filter package processing (by virus/spam filter package 1), with the opt-in value specified by the value of the attribute. The <i>Sun Schema Reference Manual</i> recommends using the <code>mailAntiUBEService</code> attribute.
ldap_optinN		UG	(New in JES MS 6.2.) The presence in a user entry of the attribute named by this option normally (but see the spamfilterN_null_optin MTA option) causes messages addressed to this user to be "opted-in" for virus/spam filter package processing (by virus/spam filter package # N), with the opt-in value specified by the value of the attribute. The value of N can

Direct LDAP attribute name MTA options

			range from 1 to 8. Note that the <i>Sun Schema Reference Manual</i> recommends using the <code>mailAntiUBEService</code> attribute for <code>optin</code> use.
<code>ldap_presence</code>		UG	RESTRICTED: Not yet used.
<code>ldap_autosecretary</code>		UG	RESTRICTED: Not yet used.
<code>ldap_alterate_recipient</code>		UG	(New in MS 8.0.1) Specify an attribute whose value contains alternate recipient address(es) to whom to send the message if it cannot be delivered to this primary recipient.
<code>ldap_start_date</code>	<code>vacationStartDate+</code>	UG	The value for this attribute should have the format <code>YYMMDDHHMMSSZ</code> , which note is in the GMT timezone. 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.
<code>ldap_end_date</code>	<code>vacationEndDate+</code>	UG	The value for this attribute should have the format <code>YYMMDDHHMMSSZ</code> , which note is in the GMT timezone. 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.
<code>ldap_conversion_tag</code>	<code>mailConversionTag</code>	UG	
<code>ldap_detourhost_optin</code>		UG	Opt-in to "detour" routing, as specified by the <code>aliasoptindetourhost</code> source channel option
<code>ldap_blocklimit</code>	<code>mailMsgMaxBlocks</code>	UG	The maximum size, in MTA blocks (see the <code>block_size</code> MTA option), of message that may be sent to a user. New in JES MS 6.3, this attribute will also (for messages that have no return-of-content policy flag already) cause messages sent from this user that are larger than the specified size to automatically get the non-return-of-content NOTARY flag set, to make it more likely that the user will be able to receive any bounce notifications about such message.
<code>ldap_mailhost</code>	<code>mailHost</code>	UG	Normally, only the host specified by this attribute may interpret (act on) a user's delivery options; however, in the case where all such delivery options are "host-independent", as on an MTA that delivers via LMTP to "back end" message store systems, or when a user entry only contains some particular delivery options that happen to be host-independent, then processing can continue even on other hosts. This attribute is optional for groups and mailing lists. If present for a group or mailing list, it specifies that that host and <i>only</i> that host can expand the group or list; if absent, any host can expand the group or list. For a user for whom a <code>mailHost</code> is required (such as a user with <code>mailbox</code> delivery option set, when <code>mailbox</code> delivery is host-dependent per <code>delivery_options</code> , with no domain level <code>ldap_domain_attr_default_mailhost</code> attribute value set), absence of a <code>mailHost</code> attribute will cause a temporary alias expansion error: 4.0.0 temporary error returned by alias expansion: address (or whatever text is configured via the <code>error_text_alias_temp</code> MTA option), the same sort of error that would occur if an LDAP problem had occurred during the lookup of the user entry (after an LDAP lookup of the domain had already succeeded).
<code>ldap_disk_quota</code>	<code>mailQuota</code>	U	Subsequent to the initial release of JES MS 6.2, support was added for <code>mailQuota</code> values specified in units other than bytes; that is, suffix characters K (kilobytes), M (megabytes), and G (gigabytes) are supported
<code>ldap_message_quota</code>	<code>mailMsgQuota</code>	U	
<code>ldap_program_info</code>	<code>mailProgramDeliveryInfo+</code>	UG	
<code>ldap_delivery_file</code>	<code>mailDeliveryFileURL</code> , <code>mailDeliveryFile</code>	UG	
<code>ldap_spare_1</code>		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>§E1</code> substitution as well as in <code>Sieve extlists</code> callouts.
<code>ldap_spare_2</code>		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>§E2</code> substitution as well as in <code>Sieve extlists</code> callouts.
<code>ldap_spare_3</code>		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>§E3</code> substitution and in <code>Sieve extlists</code> callouts.
<code>ldap_spare_4</code>		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>§E4</code> substitution and in <code>Sieve extlists</code> callouts. Note that new in JES MS 6.3, the MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines will use the preference noted in the envelope From user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_spare_5</code>		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>§E5</code> substitution. As of JES MS 6.3, the MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines will use the value tagged as being in the language of the envelope From

Direct LDAP attribute name MTA options

			user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_spare_6</code>		UGD	(New in 7.0-3.01) Specify an attribute that may be then be accessed in LDAP URL lookups via a <code>\$(E6)</code> substitution The MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope <code>From</code> user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_autoreply_mode</code>	<code>mailAutoReplyMode+</code>	UG+++	Supported values for this attribute are <code>echo</code> and <code>reply</code> . These modes will appear in a Sieve script as nonstandard <code>:echo</code> and <code>:reply</code> arguments to the <code>vacation</code> action. <code>echo</code> will produce a "processed" message disposition notification (MDN) that contains the original message as returned content. <code>reply</code> will produce a pure reply containing only the reply text. An illegal value won't manifest as any argument to the vacation action and this will produce an MDN containing only the headers of the original message.
<code>ldap_autoreply_subject</code>	<code>mailAutoReplySubject+</code>	UG+++	This attribute is used to specify the contents of the subject field to use in the vacation (autoreply) response. The value in the attribute must be a UTF-8 string. This value gets passed as the <code>:subject</code> argument to the <code>vacation</code> action. As of JES MS 6.2p2, the special strings <code>\$(SUBJECT)</code> and <code>\$(FROM)</code> are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated string. Note that new in JES MS 6.3, the MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope <code>From</code> user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_autoreply_text</code>	<code>mailAutoReplyText+</code>	UG+++	This attribute is used to store the vacation (autoreply) text (the "reason" string) returned to all senders except users in the recipient's domain. If the recipient's LDAP entry does not have a value specified for this attribute, then external users receive no vacation message. As of JES MS 6.2p2, the special strings <code>\$(SUBJECT)</code> and <code>\$(FROM)</code> are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated string. Note that new in JES MS 6.3, the MTA supports the use of multiple, language-tagged values, for this attribute;. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope <code>From</code> user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_autoreply_text_internal</code>	<code>mailAutoReplyTextInternal+</code>	UG+++	This attribute is used to store the vacation (autoreply) text (the "reason" string) returned to all senders in the recipient's own domain. If the recipient's LDAP entry does not have a value specified for this attribute, then internal users receive the external vacation text, stored in the <code>ldap_autoreply_text</code> MTA option. As of JES MS 6.2p2, the special strings <code>\$(SUBJECT)</code> and <code>\$(FROM)</code> are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated string. Note that new in JES MS 6.3, the MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as <code>Accept-Language</code> ; or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope <code>From</code> user's <code>ldap_preferred_language</code> (normally <code>preferredLanguage</code>) attribute's value.
<code>ldap_autoreply_addresses</code>		UG+++	(New in 6.2p5.) This attribute takes multiple values specifying additional addresses to recognize as "one's own" for purposes of whether to generate a vacation message. That is, it is an analogue of the <code>:addresses</code> argument for the <code>Sieve vacation</code> action.
<code>ldap_autoreply_timeout</code>	<code>mailAutoReplyTimeOut+</code>	UG+++	This attribute stores the duration, in hours, for successive vacation (autoreply) responses to any given mail sender. Used only when <code>mailAutoReplyMode=reply</code> . If the attribute's value is 0, then a response is sent back every time a message is received. This value will be converted to the nonstandard <code>:hours</code> argument to the <code>vacation</code> action. If this attribute doesn't appear on a user entry, then a default timeout will be obtained from the user's domain (from the attribute named by the <code>ldap_domain_attr_autoreply_timeout</code> MTA option) if the domain has its own timeout, or otherwise from the <code>autoreply_timeout_default</code> MTA option.
<code>ldap_filter</code>	<code>mailSieveRuleSource+</code>	UG	This attribute stores a per-user <code>Sieve filter</code> .
<code>ldap_parental_controls</code>		UG	(New in 6.2.) Specifies the name of a user or group LDAP attribute whose value can request "head of household" (a.k.a "parental controls") Sieve

Direct LDAP attribute name MTA options

			filtering be applied to this user's (or group's) messages. Any of the values <code>Yes</code> , <code>1</code> , or <code>true</code> is considered to be requesting parental controls .
ldap_filter_reference		UG	(New in 6.2.) If parental controls are enabled for a user (see the ldap_parental_controls MTA option), then the attribute named by this ldap_filter_reference MTA option specifies the DN of the entry that contains the actual head of household filter (typically, that is, the DN of the head of household user). (The attribute within that user entry containing the filter is specified by the ldap_hoh_filter MTA option, which defaults to <code>mailSieveRuleSource</code> . The lookup requests both the filter, contained in the attribute named by the ldap_hoh_filter MTA option, and the owner, contained in the attribute named by the ldap_hoh_owner MTA option, which defaults to <code>mail</code> .)
ldap_forwarding_address	<code>mailForwardingAddress+</code>	UG	Address(es) used in the expansion of named delivery options with the special value <code>*</code> (normally the name <code>forward</code> is associated with this value).
ldap_list_id	<code>mgrpUniqueId</code>	G	(New in 7.3-11.01) Single valued attribute specifying a unique identifier for the group. This identifier is used to implement MAILSERV group membership; it provides the identifier on one side of the linkage between groups and entries in the mluser tree. If this attribute is present, and the ldap_mluser_basedn MTA option is set, various virtual attributes are to the group automatically; the specifics vary depending on list policy settings.
ldap_reprocess	<code>mailDeferProcessing</code>	UG	This attribute allows per-group or per-list override of the defer_group_processing MTA option. Valid values are <code>Yes</code> , <code>No</code> , or (new in JES MS 6.3p1) <code>AFTER_AUTH</code> . <code>AFTER_AUTH</code> causes LDAP attribute based access checks, such as mgrpAllowedBroadcaster , <i>etc.</i> , to get performed "in-line", while deferring membership expansion (as well as a second check of the LDAP attribute access checks) to the reprocess channel . New in Messaging Server 7.0u3, a channel name (e.g., <code>process_special</code> or some similar, special, <code>reprocess_*</code> or <code>process_*</code> channel variant value) may be specified, and in this case the group or list expansion will be deferred to the specified channel.
ldap_jettison_domain	<code>mgrpJettisonDomain</code>	G	(New in 7.3-11.01) Messages from these domains are silently discarded. Glob-style wildcards may be used. Multiple attributes and multiple values are allowed.
ldap_jettison_url	<code>mgrpJettisonBroadcasters</code>	G	(New in 7.3-11.01) URL identifying mail addresses whose messages should be jettisoned if sent to this group. Multiple attributes and multiple values are allowed; mailto: URLs are acceptable. Each URL is expanded into a list of addresses and each address is checked against the current envelope from address. A match marks the message to be jettisoned and bypasses all other group checks and expansion. Substitution processing will be performed on this URL if bit 6 (value 64) of the process_substitutions MTA option is set.
ldap_reject_action	<code>mgrpMsgRejectAction</code>	G	(New in 6.0) Single valued attribute that controls what happens if any of the subsequent access checks fail. Only one value is defined: <code>TOMODERATOR</code> , which if set instructs the MTA to redirect any access failures to the moderator specified by the <code>mgrpModerator</code> attribute. The default (and any other value of this attribute) causes an error to be reported and the message rejected.
ldap_reject_text	<code>mgrpRejectText</code> , <code>mgrpMsgRejectText</code>	G	The name of the attribute used to store a value specifying the error text to use when an attempted posting to the group/list encounters an access failure. Because the error text may appear in SMTP responses, it must conform to SMTP response limitations. In particular, it may consist merely of a single line of text limited to the US-ASCII charset. (If the value contains eight bit characters, the entire value will be ignored. If the value contains more than a single line of text, only the first line of text will be used.)
ldap_auth_policy	<code>mgrpBroadcasterPolicy</code>	G	Specifies level of authentication needed to send to the group. Possible tokens are <code>SMTP_AUTH_REQUIRED</code> or <code>AUTH_REQ</code> , both of which mean that the SMTP AUTH command must be used to identify the sender in order to send to the group and any address produced by authentication will be used in subsequent authentication checks, <code>SMTP_AUTH_USED</code> or <code>AUTH_USED</code> , both of which force the use of any authenticated address in authorization checks but does not actually require authentication, <code>PASSWORD_REQUIRED</code> , <code>PASSWD_REQUIRED</code> , or <code>PASSWD_REQ</code> , all of which mean the password to the list specified by the <code>mgrpAuthPassword</code> attribute (see below) must appear in an Approved: header field in the message, <code>OR</code> , which changes the or_clauses MTA option setting to 1 for this list, <code>AND</code> , which changes the or_clauses MTA option setting to 0 for this list, and <code>NO_REQUIREMENTS</code> , which is basically a no-op. <code>OR</code> and <code>AND</code> are new in 6.1. This attribute is limited to a single value prior to 6.1; in 6.1 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. New in 7.3-11.01 are five new tokens for this attribute which specify list behavior for mailserv-maintained lists. The tokens are:

Direct LDAP attribute name MTA options

			<p>"LIST_OPEN", "LIST_MEMBERS", "LIST_MODERATE_NONMEMBERS", "LIST_MODERATE_MEMBERS", and "LIST_MODERATE".</p> <p>Also new in 7.3-11.01, multiple attributes can now be mapping to this attribute slot.</p>
ldap_cant_url	<code>mgrpDisallowedBroadcaster</code>	G	<p>URLs 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. Substitution processing will be performed on this URL if bit 0 (value 1) of the process_substitutions MTA option is set in 6.3 or later.</p>
ldap_auth_url	<code>mgrpAllowedBroadcaster</code>	G	<p>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. As of JES MS 6.0 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. Substitution processing will be performed on this URL if bit 1 (value 2) of the process_substitutions MTA option is set in JES MS 6.3 or later.</p> <p>New in JES MS 6.3, this now checks for address aliases (e.g., <code>mailAlternateAddress</code> and <code>mailEquivalentAddress</code>) as well as for the canonical address (normally the <code>mail</code> attribute value).</p>
ldap_cant_domain	<code>mgrpDisallowedDomain</code>	G	<p>Domains not allowed to submit messages to this group. A match means access checking has failed and all subsequent checks are bypassed. In JES MS 6.0 this check is bypassed if the submitter has already matched an ldap_auth_url. Can be multivalued and as of JES MS 6.2 glob-style wildcards are allowed.</p>
ldap_auth_domain	<code>mgrpAllowedDomain</code>	G	<p>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. In JES MS 6.0 this check is bypassed if the submitter has already matched an ldap_auth_url. As of JES MS 6.2, the value of the attribute supports use of the asterisk character, *, as a wildcard. For instance, *.domain.com means to allow all subdomains of domain.com, though not domain.com itself; to allow domain.com and all its subdomains, use two values for the attribute, domain.com and *.domain.com.</p>
ldap_maximum_message_size	<code>mgrpMsgMaxSize</code>	G	<p>Maximum message size in bytes that can be sent to the group. This attribute is obsolete but still supported for backwards compatibility; the new mailMsgMaxBlocks attribute should be used instead.</p>
ldap_auth_password	<code>mgrpAuthPassword</code>	G	<p>Specifies a password needed to post to the group.</p> <p>In iMS 5.2 the value of this attribute was saved if the mgrpBroadcasterPolicy attribute was set to require a password (see above) and checked against the Approved: field once the header is available. The Approved: field was removed from the header once the check is complete. But this did not allow for routing to the moderator in the event of a password check failure.</p> <p>In the M 6.0 release and later the presence of a mgrpAuthPassword 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>
ldap_moderator_url	<code>mgrpModerator</code>	G	<p>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 group's mgrpMsgRejectAction LDAP attribute (more precisely, the LDAP attribute named by the ldap_reject_action MTA option). If mgrpMsgRejectAction is set to "TOMODERATOR", then this attribute specifies the moderator address(es) the message is to be sent to should any of the access checks fail. If mgrpMsgRejectAction 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 isn't a 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 RFC 822 addresses or DNSs associated with the group is cleared, and the delivery options for the group are set to "members". Finally, subsequent group attributes listed in this table are ignored. Substitution processing will be performed on this URL if bit 2 (value 4) of the process_substitutions MTA option is set in 6.3 or later.</p>

Direct LDAP attribute name MTA options

<code>ldap_group_last_access_time</code>		G	<p>(New in 8.0) Specify the name of an LDAP attribute used to keep track of the last access time for email groups defined in LDAP. If this attribute is present in a group's LDAP entry, then the MTA will update the attribute each time the group is successfully accessed for purposes of sending mail or expanding a mailing list. RFC 3339 format (a profile of ISO 8601 format) is used, e.g., "2013-09-29T17:38:52Z".</p> <p>In order to prevent excessive LDAP writes, the attribute is read prior to writing and a write is only done if the current time exceeds the stored time by at least 30 minutes. (A write is also done if the attribute does not contain a valid RFC 3339 time, making it possible to set the initial value to something like "<never accessed>".)</p>
<code>ldap_group_url1</code>	<code>mgrpDeliverTo</code>	G	<p>List of URLs which, when expanded, provides a list of mailing list member addresses. Substitution processing will be performed on this URL. If bit 3 (value 8) of the <code>process_substitutions</code> MTA option is set in 6.3 or later. See also the <code>ldap_url_result_mapping</code> MTA option; with it, a mapping table can be used to manipulate the value(s) of the <code>ldap_group_url1</code> attribute.</p>
<code>ldap_group_url2</code>	<code>memberURL</code>	G	<p>Another list of URLs which, when expanded, provides another list of mailing list member addresses. Substitution processing will be performed on this URL. If bit 4 (value 16) of the <code>process_substitutions</code> MTA option is set in 6.3 or later. See also the <code>ldap_url_result_mapping</code> MTA option; with it, a mapping table can be used to manipulate the value(s) of the <code>ldap_group_url2</code> attribute.</p>
<code>ldap_group_dn</code>	<code>uniqueMember</code>	G	<p>List DNs or other identifiers for group members. Normally DNs are specified but other sorts of identifiers may be used depending on the URL template that is chosen. DNs may specify an entire subtree. These values are expanded by embedding them in an LDAP URL. The exact template to use is specified by the <code>group_dn_template</code> MTA option. The default value for this option is "ldap://\$A?mail?sub?(mail=*)"; \$A specifies the point where the <code>uniqueMember</code> DN is inserted.</p> <p>As of 7.0.5, if a <code>GROUP_TEMPLATES</code> mapping table exists, it is used as a source for the template. The mapping probe is of the form "attribute-name attribute-value". If the mapping sets \$Y, then the mapping result is used as the template instead of the <code>group_dn_template</code> MTA option value.</p> <p>Multiple values are supported but only one attribute of this type is allowed on any given group.</p> <p>As of 7.0.5 any mapping specified by the attribute named by the <code>ldap_url_result_mapping</code> MTA option will also be applied to the results produced by these attributes.</p>
<code>ldap_group_dn2</code>		G	<p>(New in 7.0.5.) Like <code>ldap_group_dn</code>, a list of DNs or other identifiers for group members. This second slot with the same semantics was added so that a single group can be defined using multiple attribute values with different semantics.</p>
<code>ldap_group_rfc822</code>	<code>mgrpRFC822MailMember</code> , <code>rfc822MailMember</code>	G	<p>Mail addresses of members of this list. Multiple values are allowed. <code>rfc822MailMember</code> is also supported for backwards compatibility with NMS, but only one of these attributes can be used in any given group.</p>
<code>ldap_url_result_mapping</code>		G	<p>(New in JES MS 6.3.) The name of the attribute used to store the name of a mapping table to be applied to the values returned from the LDAP attributes named by the <code>ldap_group_url1</code>, <code>ldap_group_url2</code>, and as of 7.0.5, the <code>ldap_group_dn</code> and <code>ldap_group_dn2</code> MTA options. That is, <code>ldap_url_result_mapping</code> specifies the name of an LDAP attribute whose value is the name of a mapping to apply to the results of expanding these attribute values. The mapping probe will be of the form:</p> <p><code>LDAP-URL LDAP-result</code></p> <p>where <code>LDAP-URL</code> is the (literal string) value of the expansion attribute, and <code>LDAP-result</code> is the value returned from LDAP for that <code>LDAP-URL</code> query. If the mapping returns with \$Y set, then the mapping result string will replace the LDAP result for alias processing purposes. If the mapping returns with \$N set, then the result will be skipped.</p> <p>This mechanism can be used to define groups based on attributes that don't contain proper email address.</p>
<code>ldap_errors_to</code>	<code>mgrpErrorsTo</code>	G	<p>Used to set an envelope From address which will override the original message's envelope From address; that is, this address is a mailing list. Setting a value for this attribute implies mailing list, rather than group, semantics: in particular, this has implications for notification messages regarding the list definition (e.g., syntactic errors in list member addresses, or syntactic errors in a list-specific Sieve filter) or regarding delivery of messages to list members, and for the handling of delivery receipt requests. Typically, the value will be some normal address. But two (three, as of 7.0-0.04) special syntaxes are also supported.</p> <p>Setting the value to an address of the form <code>user+*@domain</code> has a special meaning. The asterisk character will be expanded into a representation of the recipient address; thus a separate copy of the list message is generated for each recipient, with each copy including the intended recipient address as a subaddress within the return address. If delivery errors subsequently occur, the subaddress will indicate which was the failing</p>

Direct LDAP attribute name MTA options

			<p>address. In some cases, when dealing with remote MTAs that generate nonstandard, uninformative delivery error messages, this can in theory be useful as a way of determining which recipient address(es) failed, even when the bounce message's inner content is relatively uninformative. And it may make processing of such bounce messages by an automated program more convenient. However, the tradeoff is that such per-user-specific return address values require that a separate message copy be generated and sent for each recipient; for a "large" list, with many recipients in the same destination domains, this can be a large increase in overhead (a large decrease in efficiency). And with more prevalent use nowadays of standard format notification messages, the "need" for this sort of approach, with its extra (potentially large) overhead, is much less (since the intended recipient information can instead be extracted from the standard field in the contents of a standard format notification message).</p> <p>(New in JES MS 6.3, but not working until fix for 12194452 [Sun 6530591].) Setting the value to the forward slash character, /, has a special meaning. It tells the MTA to revert to using the original envelope From: address that had been present on the incoming message, yet in all other respects use mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender.</p> <p>New in JES MS 6.3, the <code>process_substitutions</code> MTA option can enable use of the <code>\$\$</code> (recipient's subaddress) substitution in the value. This would tend to be of interest when defining a "meta-list".</p>
<code>ldap_delay_notifications</code>	<code>mgrpDelayNotifications</code>	G	(New in 7.0u3.) Should NOTIFY=DELAY be set on list messages? Supported values are yes and no, with the obvious meanings.
<code>ldap_add_header</code>	<code>mgrpAddHeader</code>	G	Specify header fields to add to messages posted to the list. Such header fields might include, for instance, the List-*: header fields suggested in RFC 2369 (URLs for Mail List Commands through Message Headers) .
<code>ldap_remove_header</code>	<code>mgrpRemoveHeader</code>	G	Specify header fields to remove from messages posted to the list. Only the field name should be specified.
<code>ldap_add_tag</code>	<i>No default; mgrpListTag recommended</i>	G	Prefix text to insert on the Subject: header line of messages to this recipient/list, analogous to the <code>alias_tag</code> alias option, the <code>[TAG]</code> named mailing list parameter , or the effect of the Sieve addtag action. As of JES MS 6.3, the vertical bar, , character should not be used in the tag text; in previous versions, the space character should not have been used in tag text, as such use would interfere with the MTA's internal mechanisms for checking whether a tag was already present.
<code>ldap_prefix_text</code>	<code>mgrpMsgPrefixText</code>	G	Insert prefix text into messages as they undergo group expansion. Prior to Messaging Server 7.0 update 3, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.0 update 3, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The attribute values are given in UTF-8; this is then converted to match the charset of the part that the text is inserted into.
<code>ldap_suffix_text</code>	<code>mgrpMsgSuffixText</code>	G	Insert suffix text into messages as they undergo group expansion. Prior to Messaging Server 7.0 update 3, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.0 update 3, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The attribute values are given in UTF-8; this is then converted to match the charset of the part that the text is inserted into.
<code>ldap_expandable</code>	<code>mgmanMemberVisibility, expandable</code>	UG	Specify the attribute(s) used to define who (in addition to the group or list owner) may view the membership of a group or list; in MTA terms, who will get the group or list expanded in response to the SMTP EXPN command. Supported values for such an attribute or attributes are <code>anyone</code> , <code>all</code> or synonymously <code>true</code> (which means that only authenticated users -- hence users who have an account and provide their password---will be able to expand the group or list), and <code>none</code> ; unrecognized values are interpreted as <code>none</code> . Note that group or list access controls (e.g., use of attributes such as <code>mgrpAllowedBroadcaster</code> , etc., or an <code>mgrpBroadcasterPolicy</code> setting of "SMTP_AUTH_REQUIRED"), also impose restrictions on who is allowed to view list membership. All applicable conditions must be met in order for group or list membership to be viewed (expanded).
MAILSERV attributes (new in 7.4-18.01)			
<code>ldap_list_name</code>	<code>mgrpListName</code>	G	Name associated with the list for administrative purposes. The value must be a UTF-8 string.
<code>ldap_list_description</code>	<code>mgrpDescription</code>	G	Textual description of the list. UTF-8 string. Multivalued and supports language-tagged attribute values.
<code>ldap_list_advertised</code>	<code>mgrpAdvertised</code>	G	Whether or not to advertise the list in the MAILSERV GUI. Possible values are yes or no.
<code>ldap_list_public_roster</code>	<code>mgrpPublicRoster</code>	G	Should the membership list be visible to anyone, or merely to members, or to <code>admins_only</code> .
<code>ldap_list_subscribe_policy</code>	<code>mgrpSubscribePolicy</code>	G	Policy for handling list subscription requests. Possible values are: <ul style="list-style-type: none"> • "immediate" - honor subscription requests immediately • "confirm" - send confirmation e-mail • "to_moderator" - require moderator approval

Direct LDAP attribute name MTA options

			• "both" - require both confirmation and moderator approval.
ldap_list_unsubscribe_policy	<code>mgrpUnsubscribePolicy</code>	G	Controls list unsubscribe policy. Same possible settings as for ldap_list_subscribe_policy .
ldap_list_trust_new_members	<code>mgrpTrustNewMembers</code>	G	
Head of household control support attributes			
ldap_hoh_filter	<code>mailSieveRuleSource</code>		(New in JES MS 6.2.) Specify an attribute to request when performing a head of household filter lookup.
ldap_hoh_owner	<code>mail</code>		(New in JES MS 6.2.) Attribute in which to find the owner of the HOH Sieve .
Schema 2 support attributes			
ldap_attr_domain1_schema2	<code>sunPreferredDomain++</code>	D	Attribute used to store the primary domain in Schema 2.
ldap_attr_domain2_schema2	<code>associatedDomain++</code>	D	Attribute used to store any secondary domains in Schema 2.
ldap_attr_domain_search_filter			Attribute in the global configuration template area (see the ldap_global_config_templates MTA option) that is used to store the domain search filter template.
Per-domain attributes			
ldap_domain_attr_basedn	<code>inetDomainBaseDn</code>	D	Domain entry attribute used to store the baseDN of domain entries.
ldap_domain_attr_alias	<code>aliasedObjectName</code>	D	Schema 1 attribute used in domain alias entries to specify the DN of the actual domain entry.
ldap_domain_attr_uplevel		D	(New in JES MS 6.3) Specify a domain level attribute used to store a domain-specific uplevel value which overrides the value of the domain_uplevel MTA option for this particular domain. Currently only bits 0 and 2 (values 1 and 4) have meaning for the named attribute's value; the other bits of the domain_uplevel MTA option remain in effect. Note that this domain level attribute is only consulted if the domain is first <i>found</i> . Thus setting bit 0 (value 1) has no effect unless bit 2 (value 4) of the domain_uplevel MTA option is set. However, with bit 0 (value 1) of domain_uplevel is set, then clearing bit 0 in the domain level attribute can disable domain uplevel matching <i>for subdomains of this particular domain</i> while domain uplevel matching is still possible for other domains.
ldap_domain_attr_canonical	<code>inetCanonicalDomainName</code>	D	Specifies the canonical domain name for domains whose member entries overlap.
ldap_domain_attr_uid_separator	<code>domainUidSeparator</code>	D	The UID separator default for users in the domain.
ldap_domain_attr_subaddress		D	(New in 8.0) Specify the name of an LDAP attribute that controls use of subaddresses in lookups of addresses in this domain.
ldap_domain_attr_routing_hosts	<code>mailRoutingHosts</code>	D	Specify the hosts that are responsible for performing routing for this domain. If this MTA is one such host, then the user address will be looked up and attributes processed. Otherwise, the address will be routed onwards: by default, just routing based on the address, but if the MTA option route_to_routing_host is set to 1 then the first <code>mailRoutingHosts</code> value will be inserted into the address as a source route (hence the rewriting routing will depend upon that host name). Note that delivery options can be marked as mail host independent, thereby meaning that processing should occur regardless of whether this MTA is one of the <code>mailRoutingHosts</code> ; see the delivery_options MTA option.
ldap_domain_attr_smarthost	<code>mailRoutingSmartHost</code>	D	If a user address is not found in the directory, then route onwards, inserting the <code>mailRoutingSmartHost</code> value into the address as a source route.
ldap_domain_attr_status	<code>inetDomainStatus</code>	D	The attribute containing the domain's overall status.
ldap_domain_attr_mail_status	<code>mailDomainStatus</code>	D	The attribute containing the domain's mail service status. Valid values for this attribute are: <code>active</code> , <code>inactive</code> , <code>deleted</code> , <code>hold</code> , <code>disabled</code> , <code>overquota</code> , and (new in JES MS 6.0) <code>unused</code> and <code>removed</code> ; other values are interpreted as <code>inactive</code> . Note that the <code>imquotacheck</code> utility is what updates <code>mailDomainStatus</code> to set it to <code>overquota</code> .
ldap_domain_attr_blocklimit	<code>mailDomainMsgMaxBlocks</code>	D	Set a domain limit for how large of message users in the domain may receive. New in JES MS 6.3, this attribute is also checked during reverse_url lookups, and will be used (for messages that have no return-of-content policy already set) to decide whether the NOTARY non-return-of-content flag should be set.
ldap_domain_attr_conversion_tag	<code>mailDomainConversionTag</code>	D	New in JES MS 6.1. Specify a per-domain attribute whose value will be applied as conversion tags for messages sent to users or groups associated with this domain.
ldap_domain_attr_source_conversion_tag		D	New in JES MS 6.2. Specify a per-domain attribute whose value will be applied as conversion tags for messages coming from users associated with this domain.
ldap_domain_attr_optin		D	A deprecated synonym for ldap_domain_attr_optin1
ldap_domain_attr_optinn		D	Specifies the names of attributes used to opt in to spam filtering at the domain level. Messages sent or received from addresses associated with the domain will be opted in to the spam filter associated with the specified slot. Currently the slot value must be between 1 and 8.
ldap_domain_attr_nosolicit		D	New in JES MS 6.2.

ldap_domain_attr_autoreply_timeout		D	Name of an attribute that specifies a default autoreply timeout for users associated with this domain.
ldap_domain_attr_default_mailhost	No default; but the Admin SDK has a <code>preferredMailHost</code> attribute, used in provisioning, that would be one possibly appropriate attribute to also use for this purpose	D	(New in JES MS 6.3) Specify a default <code>mailHost</code> to take effect for all users in this domain who do not have their own explicit <code>mailHost</code> set.
ldap_domain_attr_disk_quota		D	
ldap_domain_attr_message_quota		D	
ldap_domain_attr_filter	<code>mailDomainSieveRuleSource</code>	D	Attribute used to specify domain-level Sieve filters .
ldap_domain_attr_report_address	<code>mailDomainReportAddress</code>	D	The domain's postmaster address . This value is used as the header From: address in DSNs reporting problems associated with recipient addresses in the domain. It is also used (in certain cases) when reporting problems to users within the domain regarding errors associated with nonlocal addresses. If this attribute is not set, then in those cases the reporting address will default to <code>postmaster@domain</code> . Note that regardless of whether or not this attribute is set, there are a number of cases where the overall host's postmaster address will be used, rather than any domain-specific postmaster address.
ldap_domain_attr_catchall_address	<code>mailDomainCatchallAddress</code>	D	Specifies the name of an attribute whose value is a "catch all" address for the domain: an address to which to route any messages to addresses "in" this domain but with an unrecognized local-part.
ldap_domain_attr_catchall_mapping	No default; <code>mailDomainCatchallMapping</code> recommended	D	(New in JES MS 6.3.) Specify the name of an attribute used to specify the name of a mapping table which will be consulted when an address associated with the domain fails to match any particular user entry. The format of the mapping table probe is the same as that of the FORWARD mapping table , and is affected by any setting of the use_forward_database MTA option in the same way as the FORWARD mapping table probe is affected. If the mapping sets the <code>\$Y</code> metacharacter, then the resulting string will replace the address being processed.
ldap_domain_attr_sourceblocklimit		D	Specify a per-domain attribute, analogous to the per-user ldap_sourceblocklimit . New behavior in JES MS 6.3 is that a per-domain setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular domain can be allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.
ldap_domain_attr_source_channel		D	(New in JES MS 6.3) Specify a source channel to which to "switch" (if userswitchchannel is specified on the current source channel)
ldap_domain_attr_recipientlimit		D	Specify a per-domain attribute, analogous to the per-user ldap_recipientlimit and the recipientlimit channel option. New behavior in JES MS 6.3 is that a per-domain setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular domain can be allowed to send messages to many recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.
ldap_domain_attr_recipientcutoff		D	Specify a per-domain attribute, analogous to the per-user ldap_recipientcutoff and the recipientcutoff channel option. New behavior in JES MS 6.3 is that a per-domain setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular domain can be allowed to send messages to many recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.
ldap_domain_attr_detourhostoptin		D	(New in 7.0.5) Specify a per-domain attribute, analogous to the per-user ldap_detourhost_optin . If this attribute has the special value (if any) specified by the aliasdetourhost_null_optin MTA option, that will be considered equivalent to the domain attribute being absent.

+ User-modifiable LDAP attribute.

++ Domain map code has the specified default, not the MTA proper

+++ While the MTA in principle allows this attribute on group/mailling list entries, the typical configuration of the [delivery_options](#) MTA option disables this support; plus, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.

39.15.6.1 Access controls on LDAP attributes

The schema sets restrictions (via an ACI) on which attributes even in his or her "own" entry an end user is allowed to modify. Reassigning the MTA's interpretation of LDAP attributes via [MTA options](#) does not, itself, affect such LDAP schema restrictions; so when reassigning end-user-modifiable LDAP attributes, be sure to also update your schema ACIs correspondingly. Also, when adding new attributes to the schema (and then making them known to the MTA via [MTA options](#)), consider in each case whether or not the new attribute should be end-user-modifiable (and in some cases consider whether the new attribute should even be end-user-visible), and when appropriate set an ACI to achieve the proper effect.

39.15.6.2 Direct LDAP attribute name MTA options: **ldap_objectclass (LDAP attribute name)**

The default for the `ldap_objectclass` MTA option is `objectClass`. This LDAP attribute is valid on `user`, `group`, and `domain` entries.

39.15.6.3 Direct LDAP attribute name MTA options: **ldap_user_status (LDAP attribute name)**

The `ldap_user_status` MTA option specifies the name of a user LDAP attribute, by default `inetUserStatus`, used to store user general status. (Contrast with the [ldap_user_mail_status MTA option](#) which names an LDAP attribute, by default `mailUserStatus`, used to store specifically a user's mail status. Also consider the analogous domain LDAP attribute, by default `inetDomainStatus`, named by the [ldap_domain_attr_status](#) MTA option.)

Prior to Messaging Server 7.0, the supported values for the LDAP attribute named by the `ldap_user_status` MTA option, by default `inetUserStatus`, were a strict subset of the supported `mailUserStatus` values, and in particular the only supported values were `active`, `inactive`, or `deleted`. As of Messaging Server 7.0, for the convenience of sites that may wish to "switch" the use (in effect switch the priority order in which checking occurs) of `inetUserStatus` and `mailUserStatus`, the full set of values supported for `mailUserStatus` are also supported for `inetUserStatus`. This is not intended to encourage general, direct use of such additional values for `inetUserStatus`, but rather, as mentioned, is intended so that the priority (order of checking) of these two status settings for users can be reordered by setting them "switched":

```
msconfig> set ldap_user_status mailUserStatus
msconfig# set ldap_user_mail_status inetUserStatus
```

or in legacy configuration mode:

```
ldap_user_status=mailUserStatus
ldap_user_mail_status=inetUserStatus
```

39.15.6.4 Direct LDAP attribute name MTA options: **ldap_user_mail_status (LDAP attribute name)**

The `ldap_user_mail_status` MTA option specifies the name of a user LDAP attribute, by default `mailUserStatus`, used to store the user's status for mail purposes. (Contrast with the [ldap_user_status MTA option](#) which names an LDAP attribute, by default `inetUserStatus`, used to store the user's general status for services in addition to mail.

And consider also the [ldap_domain_attr_mail_status](#) MTA option which names an analogous domain LDAP attribute.)

For the LDAP attribute named by `ldap_user_mail_status`, valid values are `active`, `inactive`, `disabled`, `deleted`, `overquota`, `hold`, and `new` in JES MS 6.0 removed (which from the MTA point of view is equivalent to `deleted`; it exists as a distinct status for the benefit of the `commcli` user purging operation); and `new` in JES MS 6.3 `defer` and `defer-submit` (which, respectively, mean to accept all messages to the user but defer them to the [reprocess channel](#) for later delivery (re)attempts; or in the case of `defer-submit` means to accept and defer to the [reprocess channel](#) those messages coming in a `submit` channel while giving `inactive` behavior, hence normally temporary errors, for attempted submissions on any other channels); and `new` in Messaging Server 7.3-11.01 the value `deliver` which the MTA will treat as `active` for purposes of message delivery but which other components will treat as `inactive` (giving the effect that messages can be delivered, but the user can not login); any other value is treated as `inactive`.

39.15.6.5 Direct LDAP attribute name MTA options: `ldap_group_status` (LDAP attribute name)

The `ldap_group_status` MTA option specifies the name of a group LDAP attribute intended to store general group status. There is no default; (there is no such LDAP attribute pre-defined in the schema). See also the more-specific-to-email [ldap_group_mail_status](#) MTA option, normally corresponding to the `inetMailGroupStatus` LDAP attribute. And see the [ldap_domain_attr_status](#) MTA option, normally corresponding to the `inetDomainStatus` domain LDAP attribute, which sets a domain level status.

If an LDAP attribute is defined (and added to the schema) and the MTA configured to use it via the `ldap_group_status` MTA option, note that prior to Messaging Server 7.0-0.04, the MTA's supported values were a strict subset of the values supported for `inetMailGroupStatus` (more precisely, the attribute named by [ldap_group_mail_status](#)); in particular, the supported values were `active`, `inactive`, and `deleted`. New in Messaging Server 7.0-0.04, all the values supported for `inetMailGroupStatus` are supported for this attribute as well, for the convenience of sites that wish to "switch" the priority (order) in which they are checked by "switching" which attributes the MTA options `ldap_group_status` and `ldap_group_mail_status` point to.

39.15.6.6 Direct LDAP attribute name MTA options: `ldap_group_mail_status` (LDAP attribute name)

The `ldap_group_mail_status` MTA option specifies the name of a group LDAP attribute storing the group's status for e-mail purposes. The default is `inetMailGroupStatus`. (See also the [ldap_group_status](#) MTA option for defining a more general, not specific to e-mail, group status LDAP attribute. And see the [ldap_domain_attr_mail_status](#) MTA option, normally corresponding to the `mailDomainStatus` domain LDAP attribute, which sets a domain level mail status.)

In the LDAP attribute named by `ldap_group_mail_status`, the MTA supports values of `active`, `deleted`, `removed`, `disabled`, `hold`, `inactive`, and (new in Messaging Server 7.0-0.04) `defer` and `defer-submit`.

39.15.6.7 Direct LDAP attribute name MTA options: `ldap_uid` (LDAP attribute name)

The `ldap_uid` MTA option names a user or group LDAP attribute which will be the user or group identifier, by default `uid`, used for purposes such as:

- The name (or part of the name) the user uses to log in.
- The unique name (or part of a unique name) for the user's mailbox on disk.
- Often part of the DN of the user's entry in the LDAP directory.

Note that although the MTA option `ldap_uid` exists to rename/redirect the attribute used *for MTA purposes*, other components of Messaging Server such as the Message Store tend to have hard-coded use of the attribute `uid`.

Furthermore, the use of `uid` in constructing the user's unique mailbox name on disk means that attempting to change a user's `uid` tends to be quite problematic (a change breaks access to the user's old mailbox). So make every attempt to avoid changing a user's `uid` value; use some other LDAP attribute for values subject to change (such as the user's legal name) and leave `uid` as an arbitrary, immutable, identifier.

Regarding the use of the LDAP attribute named by `ldap_uid`, normally `uid`, and its valid values: As of JES MS 6.2-0.01, the MTA checks that there is only one such attribute; as of JES MS 6.3-0.15, the MTA also checks that there is only one value set for the one attribute. As of 7.0-0.04, the MTA checks that the `uid` value is no more than 128 octets; a longer value will result in the user entry being considered invalid. (This check is performed because various lower layer libraries have hard buffer limits that preclude longer `uids`.) See also the [ldap_uid_invalid_chars MTA option](#) which enforces restrictions (some required by other components such as the Message Store) on what characters are permitted in a `uid` value. See also the [ldap_domain_attr_uid_separator](#) MTA option which names a domain level LDAP attribute specifying, for addresses in that domain, what character separates the UID from the domain name.

39.15.6.8 Direct LDAP attribute name MTA options: `ldap_mlsrange` (LDAP attribute name)

RESTRICTED: Not yet used.

39.15.6.9 Direct LDAP attribute name MTA options: `ldap_capture` (LDAP attribute name)

The `ldap_capture` MTA option specifies the names of one or more user or group LDAP attributes that will be used to trigger automatic "capturing" of user or group e-mail messages. There is no default; (no pre-defined LDAP attribute for this purpose). Typically, the LDAP attribute defined for this purpose, and named by `ldap_capture`, should be set up with an [ACI so that it is not even visible, let alone modifiable, by the users themselves](#).

As of the 8.0.1 release, the attribute `mailCaptureAddress` has been added to the Messaging Server schema for use with this attribute. However, it is still not the default.

Note that the LDAP attribute(s) specified by the [ldap_domain_attr_capture](#) MTA option have similar semantics except the attribute(s) are placed in the domain rather than in the user or group entry.

The value(s) of the LDAP attribute named by `ldap_capture` should be the address(es) to which the "captured" message copies should be sent. When a user has this attribute specified

on their LDAP entry, both messages sent to them, as well as from them, will also have a "capture" copy (normally an encapsulated copy with an entirely new message envelope) sent to the specified address.

New in Messaging Server 7.4-18.01, the [capture_format_default MTA option](#) controls whether message copies generated due to use of the LDAP attribute named by `ldap_capture` default to being in DSN encapsulated format, or to being in another format such as envelope "journal" format. Also new in Messaging Server 7.4-18.01, values of the LDAP attribute may be tagged to explicitly specify the format on a per-target-address basis: for instance, the tag `;format-report` selects the usual DSN encapsulated format, whereas the tag `;format-journal` selects the envelope "journal" format. New in the 8.0 release are the attribute tags `;format-message`, `;format-report-header`, and `;format-journal-header`, which can be used to specify header-only capture addresses.

39.15.6.10 Direct LDAP attribute name MTA options: `ldap_recipientlimit` (LDAP attribute name)

The `ldap_recipientlimit` MTA option specifies the name of a user or group LDAP attribute that will be used to store a sending-user-specific maximum number of envelope recipients per message submission (additional recipients are rejected), analogous to the [recipientlimit channel option](#). There is no default; (no pre-defined LDAP attribute for this purpose).

New behavior in JES MS 6.3-0.15 is that a per-user setting such as this will override more general settings such as a channel [recipientlimit](#), rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3-0.15, a particular user can be allowed to send messages to a large number of recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect. (But a TCP/IP-channel-specific option setting of [ALLOW_RECIPIENTS_PER_TRANSACTION](#) can *not* be overridden!)

See also the [ldap_recipientcutoff](#) MTA option for a similar but slightly different effect. And see the [ldap_domain_attr_recipientlimit](#) MTA option for similar effect at the domain, rather than user, level.

39.15.6.11 Direct LDAP attribute name MTA options: `ldap_recipientcutoff` (LDAP attribute name)

The `ldap_recipientcutoff` MTA option specifies the name of a user or group LDAP attribute that will be used to store a sending-user-specific maximum number of envelope recipients per message submission (messages with more recipients are rejected entirely), analogous to the [recipientcutoff channel option](#). There is no default; (no pre-defined LDAP attribute for this purpose).

New behavior in JES MS 6.3-0.15 is that a per-user setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3-0.15, a particular user can be allowed to send messages to a large number of recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.

See also the [ldap_recipientlimit](#) MTA option for a similar but slightly different effect. And see the [ldap_domain_attr_recipientcutoff](#) MTA option for a similar effect at the domain, rather than user, level.

39.15.6.12 Direct LDAP attribute name MTA options: **ldap_sourceblocklimit (LDAP attribute name)**

The `ldap_sourceblocklimit` MTA option specifies the name of a user or group LDAP attribute used to store a sending-user-specific maximum message size, analogous to the [sourceblocklimit channel option](#). There is no default, (no pre-defined LDAP attribute for this purpose); if desiring to have the exact same sending limit as receiving limit, then the `ldap_sourceblocklimit` MTA option could be set to the same value as the [ldap_blocklimit MTA option](#), which by default is `mailMsgMaxBlocks`.

New behavior in JES MS 6.3-0.15 is that a per-user setting such as the value of the LDAP attribute named by `ldap_sourceblocklimit` will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3-0.15, a particular user can be allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.

39.15.6.13 Direct LDAP attribute name MTA options: **ldap_source_channel (LDAP attribute name)**

(New in JES MS 6.3-0.15) The `ldap_source_channel` MTA option specifies the name of a user or group LDAP attribute which will be used to store the source channel for messages submitted by this user. There is no default; (no pre-defined LDAP attribute for this purpose).

The value of the LDAP attribute named by `ldap_source_channel` sets a source channel to which to "switch" for messages submitted by the user, if [userswitchchannel](#) is set on the current source channel. If the user LDAP attribute is present, it will override any domain-level setting made via the LDAP attribute named by the [ldap_domain_attr_source_channel MTA option](#).

39.15.6.14 Direct LDAP attribute name MTA options: **ldap_source_optinN for N=1--8 (LDAP attribute name)**

(New in JES MS 6.3-0.15) The `ldap_source_optinN` MTA options specify, respectively, the names of user/group LDAP attributes used to select "opt in" to spam/virus filter package N, N=1--8. These MTA options have no default (no pre-defined LDAP attribute for the purpose). If it is desired to have "opt in" for the messages a user sends, just the same as for the messages a user receives, then both `ldap_source_optinX` and `ldap_optinX` could be set to the same LDAP attribute.

The LDAP attributes named by `ldap_source_optinN` are the sending user analogues of [ldap_optinN](#). In particular, the presence in a user entry of the LDAP attribute named by `ldap_source_optinN` normally (but see the [spamfilterN_null_optin MTA options](#)) causes messages sent *from* that user to be "opted-in" for spam/virus filtering via the spam/virus filter package N, with the "opt-in" value specified by the value of the LDAP attribute.

39.15.6.15 Direct LDAP attribute name MTA options: **ldap_preferred_language (LDAP attribute name)**

The `ldap_preferred_language` MTA option specifies the name of a user or group LDAP attribute, by default `preferredLanguage`, used to store the user's language preference.

The schema sets an ACI on the default attribute, `preferredLanguage`, to allow user modification.

The MTA's typical [NOTIFICATION_LANGUAGE mapping table](#) and [DISPOSITION_LANGUAGE mapping table](#) check the value of this attribute (for the sender of the original message) when deciding in what language to send back notification messages. Also, as of JES MS 6.3-0.15, the MTA has the ability to choose between multiple LDAP attribute values with different language tags and determine the preferred value to use. The language tags in effect are compared against the preferred language information associated with the envelope From address. In JES MS 6.3-0.15, the only attributes receiving this treatment are those named by [ldap_autoreply_subject](#) (normally `mailAutoReplySubject`), [ldap_autoreply_text](#) (normally `mailAutoReplyText`), [ldap_autoreply_text_internal](#) (normally `mailAutoReplyTextInternal`), [ldap_autoreply_addresses](#), [ldap_prefix_text](#), [ldap_suffix_text](#), [ldap_spare_4](#), and [ldap_spare_5](#). As of Messaging Server 7.0-3.01, the attribute named by (new in that version) [ldap_spare_6](#) also receives such treatment. As of Messaging Server 7.2-7.02, any of the [ldap_spare_N](#) named attributes may optionally, depending upon the setting of the corresponding [spare_N_separator MTA option](#), receive `preferredLanguage` treatment; the default for the `spare_N_separator` MTA options is such that the `ldap_spare_4`, `ldap_spare_5`, `ldap_spare_6` named attributes receive `preferredLanguage` treatment. As of Messaging Server 7.3-11.01, the attribute named by [ldap_add_tag](#) also receives such treatment.

39.15.6.16 Direct LDAP attribute name MTA options: `ldap_preferred_country` (LDAP attribute name)

The `ldap_preferred_language` MTA option specifies the name of a user or group LDAP attribute used to store a country preference; there is no default.

39.15.6.17 Direct LDAP attribute name MTA options: `ldap_nosolicit` (LDAP attribute name)

The `ldap_nosolicit` MTA option specifies the name of a user or group level LDAP attribute intended for users or groups to specify what classes of e-mail solicitations they wish to reject. See [RFC 3865 \(NO-SOLICITING SMTP Extension\)](#) for a discussion of the SMTP extension utilized.

The `ldap_nosolicit` MTA option has no default: there is no LDAP attribute already in the schema for this purpose. If adding some LDAP attribute to the schema for this purpose, consider establishing it with an ACI allowing users to modify their own attribute's value themselves.

Note that the MTA expects the value of whatever attribute is named by `ldap_solicit` to consist of a comma-separated list of strings.

39.15.6.18 Direct LDAP attribute name MTA options: `ldap_routing_address` (LDAP attribute name)

The `ldap_routing_address` MTA option specifies the name of a user or group LDAP attribute, by default `mailRoutingAddress`, used to specify an address to which to route, overriding (as of JES MS 6.0) the usual [mailHost](#) and [mailDeliveryOption](#) interpretation.

39.15.6.19 Direct LDAP attribute name MTA options:

ldap_delivery_option (LDAP attribute name)

The `ldap_delivery_option` MTA option specifies the name of a user or group LDAP attribute, by default `mailDeliveryOption`, used to specify the delivery choices of the user or group. See the [delivery_options](#) MTA option for discussion of the MTA's interpretation of the values stored in this named attribute.

The schema sets an ACI on the default attribute, `mailDeliveryOption`, to allow user modification.

39.15.6.20 Direct LDAP attribute name MTA options:

ldap_personal_name (LDAP attribute name)

The `ldap_personal_name` MTA option specifies the name of a user or group LDAP attribute used to specify a user's (or group's) choice of personal name. This option has no default; some sites might choose to set it to `cn`.

If the `ldap_personal_name` MTA option is set, then the value of the specified attribute (if present in a user entry) will be inserted by the MTA as a personal name wherever the user's address appears in message headers (overriding any originally present personal name for the user that might have been present), including when generating vacation messages on behalf of the user. Note that (as of 6.2p3 for normal messages, or as of 6.2p6 for generated messages such as vacation messages) the MTA will quote the value obtained from LDAP, if required according to the quoting rules for personal names (technically "phrases") given in [RFC 822](#).

For messages submitted through MSHTTP (messages submitted from web clients), see also the [fullfromheader](#) MSHTTP option.

39.15.6.21 Direct LDAP attribute name MTA options:

ldap_source_conversion_tag (LDAP attribute name)

The `ldap_source_conversion_tag` MTA option specifies the name of a user or group LDAP attribute which may be used to specify a [conversion tag](#) to add to messages sent from that user or group. The option has no default; sites that wish to use exactly the same conversion tag(s) for messages *from* as well as *to* a user might set it to `mailConversionTag`, while sites wishing to distinguish directionality in conversion tags will wish to use a distinct LDAP attribute.

Note that there is a domain level analogue, [ldap_domain_attr_source_conversion_tag](#).

39.15.6.22 Direct LDAP attribute name MTA options:

ldap_primary_address (LDAP attribute name)

The `ldap_primary_address` MTA option specifies the name of the user or group LDAP attribute which contains the primary email address for that user or group, by default the `mail` attribute.

Compare with the MTA options that specify the names of LDAP attributes which store email aliases, [ldap_alias_addresses](#) which by default names the `mailAlternateAddress`

attribute (or in the `sims40` schema, names the `rfc822mailalias` attribute), and [ldap_equivalence_addresses](#) which by default names the `mailEquivalentAddress`.

39.15.6.23 Direct LDAP attribute name options: `ldap_alias_addresses` (list of LDAP attribute names)

The `ldap_alias_addresses` MTA option specifies the name(s) of the user or group LDAP attribute(s) in which e-mail aliases will be stored. That is, this MTA option names LDAP attributes whose values (addresses) are accepted as equivalent to (aliases for) the canonical mail address on incoming messages; see also the [ldap_mail_reverses](#) MTA option which controls just which attributes (addresses) are normally converted to the canonical mail address during [reverse_url](#) application via the [\\$Q substitution sequence](#). Use a comma-separated list if specifying more than one LDAP attribute name.

The default for `ldap_alias_addresses` depends upon the schema tag in effect, as set via the [ldap_schematag](#) MTA option. Normally, with `ldap_schematag=ims50` set (or with `ldap_schematag=nms41` set), the default for `ldap_alias_addresses` is `mailAlternateAddress`. But if `ldap_schematag=sims40` is set, then the default is instead `ldap_alias_addresses=rfc822mailalias`.

The aliases stored in a `ldap_alias_addresses` LDAP attribute are subject to address reversal (canonicalization back to the value of the LDAP attribute named by [ldap_primary_address](#), normally the `mail` attribute) by the MTA. In contrast, for an alias intended to be emitted as well as recognized, see the [ldap_equivalence_addresses](#) MTA option, normally naming the `mailEquivalentAddress` LDAP attribute.

39.15.6.24 Direct LDAP attribute name options: `ldap_equivalence_addresses` (list of LDAP attribute names)

The `ldap_equivalence_addresses` MTA option specifies the name(s) of the user or group LDAP attribute(s) in which e-mail aliases will be stored. That is, this MTA option names the LDAP attributes containing addresses accepted as equivalent to the canonical mail address for incoming messages; such equivalent addresses are also allowed to appear on outgoing messages (are not converted during [reverse_url](#) application). (In contrast, for aliases that will be accepted but canonicalized, see the [ldap_alias_addresses](#) MTA option.) Use a comma-separated list if specifying more than one LDAP attribute name.

Note that when setting the `ldap_equivalence_addresses` MTA option to a non-default value, it is also usually appropriate/necessary to modify the [ldap_mail_aliases](#) MTA option correspondingly (to include those attribute(s) named by `ldap_equivalence_addresses`).

39.15.6.25 Direct LDAP attribute name options: `ldap_optinN` (list of LDAP attribute names)

The `ldap_optinN`, $N=1, \dots, 8$, MTA options may be used to name user or group LDAP attributes whose presence in an entry normally (but see the [spamfilterN_null_optin](#) MTA options) causes messages addressed to that user or group to be "opted-in" for [spam/virus filter package processing](#) (by spam/virus filter package N), with the opt-in value specified by the value of the attribute.

These MTA options have no default, though the *Sun One Messaging and Collaboration Schema Reference Manual* (back in the days where it was assumed that at most one spam/virus filter

package would be used) suggested using an LDAP attribute named `mailAntiUBEService`. The `mailAntiUBEService` attribute is defined in the schema. If you will be using multiple spam/virus filter packages and wish to have distinct attributes for the different spam/virus filter packages, you may define new attributes modelled on `mailAntiUBEService`, perhaps using suggestive attribute names corresponding to their function, *e.g.*, `milterOptin`, *etc.*.

For aliases defined via an [alias option](#), the analogous option is `alias_optinN`; or for aliases defined in the [alias file](#), the analogous setting is the [Alias file named parameter \[OPTIN#\]](#).

39.15.6.26 Direct LDAP attribute name MTA options: `ldap_presence` (LDAP attribute name)

RESTRICTED: Not yet used.

39.15.6.27 Direct LDAP attribute name MTA options: `ldap_autosecretary` (LDAP attribute name)

RESTRICTED: Not yet used.

39.15.6.28 Direct LDAP attribute name MTA options: `ldap_alternate_recipient` (list of LDAP attribute names)

(New in MS 8.0.1.) The `ldap_alternate_recipient` MTA option specifies the name(s) of one or more user-level LDAP attributes whose value(s) are alternate recipient addresses to whom to send messages that cannot be delivered to this primary recipient.

Compare with the [alias option](#) `alias_alternate_recipient`.

39.15.6.29 Direct LDAP attribute name MTA options: `ldap_start_date` (LDAP attribute name)

The `ldap_start_date` MTA option specifies the name of a user (or group) LDAP attribute, by default `vacationStartDate`, used to specify the start of the date/time range for which to apply autoreply (vacation) processing. (The MTA in principle allows this attribute on group/ mailing list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables interpreting the attribute's value in the case of groups. Furthermore, the Sun schema does not, as distributed, allow this attribute on group/ mailing list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/ mailing list entries.)

The value stored in the named attribute should have the format `YYYYMMDDHHMMSSZ`, which note is in the GMT timezone. An autoreply will only be generated if the current time is after the time specified by the attribute's value (and before the time specified by the value of the attribute named by `ldap_end_date`, by default `vacationEndDate`). No start date is enforced if the attribute named by `ldap_start_date` is missing. See the [delivery_options](#) MTA option for some additional discussion of limiting autoreply processing to those messages received within the `vacationStartDate` to `vacationEndDate` time range.

The schema sets an ACI on the default attribute, `vacationStartDate`, to allow user modification.

39.15.6.30 Direct LDAP attribute name MTA options: `ldap_end_date` (LDAP attribute name)

The `ldap_end_date` MTA option specifies the name of a user (or group) LDAP attribute, by default `vacationEndDate`, used to specify the end of the date/time range for which to apply autoreply (vacation) processing. (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables interpreting the attribute's value in the case of groups. Futhermore, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.)

The value stored in the named attribute should have the format `YYYYMMDDHHMMSSZ`, which note is in the GMT timezone. An autoreply will only be generated if the current time is before the time specified by the attribute's value (and after the time specified by the value of the attribute named by `ldap_start_date`, by default `vacationStartDate`). No end date is enforced if the attribute named by `ldap_end_date` is missing. See the [delivery_options](#) MTA option for some additional discussion of limiting autoreply processing to those messages received within the `vacationStartDate` to `vacationEndDate` time range.

The schema sets an ACI on the default attribute, `vacationEndDate`, to allow user modification.

39.15.6.31 Direct LDAP attribute name MTA options: `ldap_conversion_tag` (LDAP attribute name)

The `ldap_conversion_tag` MTA option specifies the name of a user or group LDAP attribute, by default `mailConversionTag`, which may be used to specify a [conversion tag](#) to add to messages addressed to that user or group.

Compare with [ldap_source_conversion_tag](#) which may be used to specify a conversion tag to add to messages sent *from*, instead of *to*, a user or group. Note that there is also a domain level analogue, [ldap_domain_attr_conversion_tag](#).

39.15.6.32 Direct LDAP attribute name MTA options: `ldap_detourhost_optin` (LDAP attribute name)

The `ldap_detourhost_optin` MTA option specifies the name of a user or group LDAP attribute which may be used to specify "opt-in" to "detour" routing, as specified by the [aliasoptindetourhost](#) channel option. Normally the presence of the specified attribute on a user or group entry causes "opt in", however, see the [aliasdetourhost_null_optin](#) MTA option.

Note that there is a domain level analogue, [ldap_domain_attr_detourhostoptin](#).

39.15.6.33 Direct LDAP attribute name MTA options: `ldap_blocklimit` (LDAP attribute name)

The `ldap_blocklimit` MTA option specifies the name of a user or group LDAP attribute, by default `mailMsgMaxBlocks`, which may be used to specify the maximum size, in MTA blocks (see the [block_size](#) MTA option), of messages that may be sent to the user or group.

New in JES MS 6.3, this attribute will also (for messages that have no return-of-content policy flag already) cause messages sent from this user that are larger than the specified size to automatically get the non-return-of-content NOTARY flag set, to make it more likely that the user will be able to receive any bounce notifications about such message.

Compare with the [alias option `alias_blocklimit`](#), with the domain-level [ldap_domain_attr_blocklimit](#) MTA option, and with the similar limit on messages a user may *send* controlled via the [ldap_sourceblocklimit](#).

39.15.6.34 Direct LDAP attribute name MTA options: `ldap_mailhost` (list of LDAP attribute names)

The `ldap_mailhost` MTA option specifies the names of user (or group) LDAP attributes, by default `mailHost`. used to specify the mail host of the user (or group). When a user (or group) has no explicitly specified `mailHost`, see also the [ldap_domain_attr_default_mailhost](#) MTA option which if set, specifies a domain level LDAP attribute for the MTA to use as a default `mailHost` for users without their own explicit value set.

Normally, only the host specified by `mailHost` attribute may interpret (act on) a user's delivery options; however, in the case where all such delivery options are "host-independent", as on an MTA that delivers via LMTP to "back end" message store systems, or when a user entry only contains some particular delivery options that happen to be host-independent, then processing can continue even on other hosts. This attribute is optional for groups and mailing lists. If present for a group or mailing list, it specifies that that host and only that host can expand the group or list; if absent, any host can expand the group or list. For a user for whom a `mailHost` is required (such as a user with `mailbox` delivery option set, when `mailbox` delivery is host-dependent per [delivery_options](#), with no domain level [ldap_domain_attr_default_mailhost](#) attribute value set), absence of a `mailHost` attribute will cause a temporary alias expansion error:

```
452 4.0.0 temporary error returned by alias expansion: address
```

(or whatever text is configured via the [error_text_alias_temp](#) MTA option), the same sort of error that would occur if an LDAP problem had occurred during the lookup of the user entry (after an LDAP lookup of the domain had already succeeded).

For some further discussion of the use of `mailHost`, see the [Overview of Direct LDAP](#) discussion.

For the MMP, see the analogous [mailhostattrs](#) option; for authentication results, see the similar [ldap_auth_attr_mail_host](#) option.

39.15.6.35 Direct LDAP attribute name MTA options: `ldap_disk_quota` (LDAP attribute name)

The `ldap_disk_quota` MTA option names the user LDAP attribute used to set user disk quota, by default the `mailQuota` LDAP attribute. See also the [ldap_message_quota MTA option](#), which names an LDAP attribute storing a per-message quota (size limit).

By default, the value stored in the LDAP attribute named by `ldap_disk_quota` is assumed to have units of bytes. Suffix characters on the value allow units of other than bytes: `K` (kilobytes), `M` (megabytes), or `G` (gigabytes) are supported.

39.15.6.36 Direct LDAP attribute name MTA options: `ldap_message_quota` (LDAP attribute name)

The `ldap_message_quota` MTA option names the user LDAP attribute, by default `mailMsgQuota`, used to set a user per-message quota (size limit). See also the [ldap_disk_quota MTA option](#), which names an LDAP attribute storing user disk quota.

39.15.6.37 Direct LDAP attribute name MTA options: `ldap_program_info` (LDAP attribute name)

The `ldap_program_info` MTA option specifies the name of a user or group LDAP attribute, by default `mailProgramDeliveryInfo`, whose value specifies the program which the [pipe channel](#) will run to effect delivery for a [mailDeliveryOption](#) value of `program`. (A `mailDeliveryOption` value of `program` causes routing to the pipe channel per the `program` clause of the [delivery_options](#) MTA option's value; then the pipe channel interprets the value of `mailProgramDeliveryInfo` to determine how to perform the actual delivery.)

The schema sets an ACI on the default attribute, `mailProgramDeliveryInfo`, to allow user modification.

39.15.6.38 Direct LDAP attribute name MTA options: `ldap_delivery_file` (LDAP attribute name list)

The `ldap_delivery_file` MTA option specifies the name of user or group LDAP attributes, by default `mailDeliveryFileURL` and `mailDeliveryFile`, whose values specify a file to which to "deliver" messages. (This might, for instance, be a list posting archive file.)

39.15.6.39 Direct LDAP attribute name MTA options: `ldap_spare_N` (LDAP attribute name)

The `ldap_spare_N`, $N=1, \dots, 18$, MTA options may be used to name LDAP attributes intended for site-customizable purposes, to be made known to the MTA (and hence be more easily accessible in MTA [LDAP URLs](#) and certain MTA [mapping tables](#), *etc.*). `ldap_spare_6` was new in Messaging Server 7.0-3.01; `ldap_spare_N` for $N=7, \dots, 18$ were new in Messaging Server 7.2-7.02.

When a `ldap_spare_N` option has been set to the name of an LDAP attribute, then the value of the named attribute may be substituted in MTA [LDAP URLs](#) via the `$NE` substitution sequence.

For $N=1, \dots, 6$, the value of the named attribute may optionally (see the [include_spare](#) MTA option) be included in various [recipient access mapping table](#) probes and [FROM_ACCESS mapping table](#) probes. And as of 8.0, such named LDAP attribute values may also be included (see the [include_spare2](#) MTA option) in [FORWARD mapping table](#) probes.

`ldap_spare_4`, `ldap_spare_5`, and `ldap_spare_6` will be included in [SIEVE_EXTLISTS mapping table](#) probes. And as of JES MS 6.3, the MTA supports the use of multiple, language-tagged values for these (`ldap_spare_4`, `ldap_spare_5`, and `ldap_spare_6`) attributes. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as

Accept-Language;, or in the absence of such header lines will use the value tagged as being in the language of the envelope From user's [ldap_preferred_language](#) (normally `preferredLanguage`) attribute's value.

See the respective [spare_N_separator](#) MTA options for configuration of whether a `ldap_spare_N` LDAP attribute is allowed to have multiple values and if so, how to handle the multiple values.

For aliases defined in the alias file (legacy configuration), or via [alias options](#) (Unified Configuration), see the [\[SPARE*\]](#) alias file named parameters or [alias_spare*](#) alias options, respectively.

39.15.6.40 Direct LDAP attribute name MTA options: `ldap_autoreply_mode` (LDAP attribute name)

The `ldap_autoreply_mode` MTA option specifies the name of the user (or group) LDAP attribute which will store what "type" of autoreply/vacation message the user or group wishes to emit. The default LDAP attribute for this purpose is `mailAutoReplyMode`. (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables support for interpreting the attribute's value in the case of groups. Futhermore, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.)

The [schema sets an ACI](#) on the default attribute, `mailAutoReplyMode`, to allow user modification.

Supported values for the value of the LDAP attribute named by `ldap_autoreply_mode`, e.g., `mailAutoReplyMode`, are `echo` and `reply`. These modes will appear in the [Sieve script](#) (constructed by the MTA on the fly, based on these LDAP values) 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 won't manifest as any argument to the vacation action and this will produce an MDN containing only the headers of the original message.

39.15.6.41 Direct LDAP attribute name MTA options: `ldap_autoreply_subject` (LDAP attribute name)

The `ldap_autoreply_subject` MTA option specifies the name of the user (or group) LDAP attribute which will store the text to put on the Subject: header line of any autoreply/vacation message generated on behalf of the user. The default LDAP attribute for this purpose is `mailAutoReplySubject`. (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables this support in the case of groups. Futhermore, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.)

The [schema sets an ACI](#) on the default attribute, `mailAutoReplySubject`, to allow user modification.

In whatever attribute is named by `ldap_autoreply_subject`, the value in the attribute must be a UTF-8 string. This value gets passed as the `:subject` argument to the `vacation` action.

As of JES MS 6.2-2.01, the special strings `$SUBJECT` and `$FROM` are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated string.

Note that new in JES MS 6.3-0.15, the MTA supports the use of multiple, language-tagged values, for this attribute. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as `Accept-Language:`, or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope From user's [ldap_preferred_language](#) (normally `preferredLanguage`) attribute's value.

Prior to 7.0.5, the length of the value of `mailAutoReplySubject` was limited to 256 characters (though the maximum length of a `:subject` parameter in a Sieve `vacation` action has always been 1024 characters). As of 7.0.5, the length limit for `mailAutoReplySubject` has been raised so that at least 900 characters can be specified.

39.15.6.42 Direct LDAP attribute name MTA options: `ldap_autoreply_text` (LDAP attribute name)

The `ldap_autoreply_text` MTA option specifies the name of the user (or group) LDAP attribute which will store the vacation/autoreply text (the "reason" string) returned to all senders except users in the recipient's own domain. The default LDAP attribute for this purpose is `mailAutoReplyText`. (The MTA in principle allows this attribute on group/ mailing list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables this support in the case of groups. Furthermore, the Sun schema does not, as distributed, allow this attribute on group/ mailing list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/ mailing list entries.) If the recipient's LDAP entry does not have a value specified for the attribute named by `ldap_autoreply_text` (normally the `mailAutoReplyText` attribute), then note that "external" senders will receive no vacation message. Note that when generating an autoreply/vacation message back to a fellow "internal" user, the value of the attribute named by the [ldap_autoreply_text_internal](#) MTA option will be used instead of the value of the attribute named by the [ldap_autoreply_text](#) MTA option.

The [schema sets an ACI](#) on the default attribute, `mailAutoReplyText`, to allow user modification.

As of JES MS 6.2-2.01, the special strings `$SUBJECT` and `$FROM` are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated body text string.

Note that new in JES MS 6.3-0.15, the MTA supports the use of multiple, language-tagged values, for this attribute;. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as `Accept-Language:`, or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope From user's [ldap_preferred_language](#) (normally `preferredLanguage`) attribute's value.)

39.15.6.43 Direct LDAP attribute name MTA options: `ldap_autoreply_text_internal` (LDAP attribute name)

d

The `ldap_autoreply_text_internal` MTA option specifies the name of the user (or group) LDAP attribute which will store the vacation/autoreply text (the "reason" string) returned to all senders in the recipient's own domain. (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables this support in the case of groups. Furthermore, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.) If the recipient's LDAP entry does not have a value specified for this attribute, then internal users receive the external vacation text, stored in the [ldap_autoreply_text](#) MTA option.

The [schema sets an ACI](#) on the default attribute, `mailAutoReplyTextInternal`, to allow user modification.

As of JES MS 6.2-2.01, the special strings `$SUBJECT` and `$FROM` are supported for use in this attribute's value, causing substitution of the original message's Subject: field value or From: field value, respectively, into the generated body text string.

Note that new in JES MS 6.3-0.15, the MTA supports the use of multiple, language-tagged values, for this attribute;. When multiple, language-tagged values are present, the MTA will preferentially use the value tagged as being in the language preference expressed in a header line such as `Accept-Language:`, or in the absence of such header lines the MTA's next choice will be the value tagged as being in the language of the envelope From user's [ldap_preferred_language](#) (normally `preferredLanguage`) attribute's value.)

39.15.6.44 Direct LDAP attribute name MTA options: `ldap_autoreply_addresses` (LDAP attribute name)

The `ldap_autoreply_addresses` MTA option specifies the name of an LDAP attribute in which to store additional, recognized-for-vacation-message-purposes, versions of the recipient's address; there is no default (no pre-defined LDAP attribute for this purpose). (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables this support in the case of groups. Furthermore, the Sun schema does not, as distributed, define an attribute for this purpose. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.) If adding such an LDAP attribute to the schema, this is an attribute to consider making user self-modifiable (have an [ACI set on it](#) allowing user modification).

The attribute named by the `ldap_autoreply_addresses` MTA option takes multiple values specifying additional addresses to recognize as "one's own" for purposes of whether to generate a vacation message. That is, it is an analogue of the `:addresses` argument for the [Sieve](#) `vacation` action. As of JES MS 6.3-0.15, the MTA supports use of language-tagged values for the attribute named by `ldap_autoreply_addresses`.

39.15.6.45 Direct LDAP attribute name MTA options: `ldap_autoreply_timeout` (LDAP attribute name)

The `ldap_autoreply_timeout` MTA option specifies the name of the user (or group) LDAP attribute which will store the duration (the "timeout") between successive vacation/autoreply

responses to any given sender. (The MTA in principle allows this attribute on group/mailling list entries as well as on user entries, but the typical configuration of the [delivery_options](#) MTA option disables this support in the case of groups. Furthermore, the Sun schema does not, as distributed, allow this attribute on group/mailling list entries. See the [delivery_options](#) MTA option for some discussion regarding enabling use of this attribute on group/mailling list entries.)

The [schema sets an ACI](#) on the default attribute, `mailAutoReplyTimeOut`, to allow user modification.

The value in the attribute named by `ldap_autoreply_timeout` should be represented in units of hours. If the attribute's value is 0, then a response is sent back every time a message is received. The attribute's value will be converted to the nonstandard argument to the vacation action. (Note that the standard [Sieve](#) vacation action is defined to only support the `:days` argument for this purpose and doesn't allow a value of 0; so this support for shorter intervals between responses is an extension in the MTA's Sieve support.)

If the attribute named by `ldap_autoreply_timeout` is not present in a user entry, then a default timeout will be obtained from the user's domain (from the attribute named by the [ldap_domain_attr_autoreply_timeout](#) MTA option) if the domain has its own timeout, or otherwise from the [autoreply_timeout_default](#) MTA option.

39.15.6.46 Direct LDAP attribute name MTA options: `ldap_filter` (LDAP attribute name)

The `ldap_filter` MTA option specifies the name of the LDAP attribute which will store the [Sieve filter](#) of the user or group.

The [schema sets an ACI](#) on the default attribute, `mailSieveRuleSource`, to allow user modification.

39.15.6.47 Direct LDAP attribute name MTA options: `ldap_parental_controls` (LDAP attribute name)

The `ldap_parental_controls` MTA option names a user or group LDAP attribute to use to select whether or not a user will have "parental controls" (or "head-of-household controls") applied. There is no default; (no pre-defined LDAP attribute for this purpose). Note that when adding an attribute for this purpose, this typically should *not* be an attribute modifiable by the user himself; instead, it should typically instead be modifiable by the "parent" ("head-of-household") user.

For the LDAP attribute named by `ldap_parental_controls`, any of the values `Yes`, `1`, or `true` is considered to be requesting parental controls. If such a value is present enabling parental controls, then the LDAP attribute named by the [ldap_filter_reference](#) MTA option will be consulted to determine who (what DN in the directory) is the "parent" of ("head-of-household" for) this user.

39.15.6.48 Direct LDAP attribute name MTA options: `ldap_filter_reference` (LDAP attribute name)

The `ldap_filter_reference` MTA option specifies the name of a user or group LDAP attribute to be used to locate the "parent" of ("head-of-household" for) the current user or

group entry. There is no default; (no pre-defined LDAP attribute for this purpose). Note that when adding an attribute for this purpose, this typically should *not* be an attribute modifiable by the user himself; instead, it should typically instead be modifiable by the "parent" ("head-of-household") user.

If parental controls are enabled for a user (see the [ldap_parental_controls MTA option](#)), then the attribute named by this [ldap_filter_reference MTA option](#) specifies the DN of the entry that contains the actual parent/head of household Sieve filter (typically, that is, the DN of the head of household user). (The LDAP attribute within that head of household user entry containing the Sieve filter is specified by the [ldap_hoh_filter MTA option](#), which defaults to `mailSieveRuleSource`. The lookup requests both the Sieve filter, contained in the attribute named by the `ldap_hoh_filter` MTA option, and the owner, contained in the LDAP attribute named by the [ldap_hoh_owner MTA option](#), which defaults to `mail`.)

39.15.6.49 Direct LDAP attribute name MTA options: `ldap_forwarding_address` (LDAP attribute name)

The `ldap_forwarding_address` MTA option specifies the name of a user or group LDAP attribute which will store a forwarding address for the user or group, to be used if `mailDeliveryOption` attribute (or whatever attribute is named by the [ldap_delivery_option MTA option](#)) has a value of `forward`.

The [schema sets an ACI](#) on the default attribute, `mailForwardingAddress`, to allow user modification.

39.15.6.50 Direct LDAP attribute name MTA option: `ldap_reprocess` (LDAP attribute name)

The `ldap_reprocess` MTA option specifies the name of a user or group LDAP attribute which will control whether alias expansion of the entry is deferred to the [reprocess channel](#) *vs.* being performed immediately (in-line). The default is `mailDeferProcessing`. The value of the attribute overrides the MTA's general default controlled by the [defer_group_processing MTA option](#).

Valid values for the LDAP attribute named by `ldap_reprocess` include: "Yes", "No", or (new in JES MS 6.3p1) "AFTER_AUTH". "AFTER_AUTH" causes LDAP attributed based access checks, such as `mgrpAllowedBroadcaster`, *etc.*, to get performed "in-line", while deferring membership expansion (as well as a second check of the LDAP attribute based access checks) to the [reprocess channel](#). New in Messaging Server 7.0u3, it is also valid to specify as the attribute's value a channel name (*e.g.*, "process_special" or the name of some similar, special, `reprocess_*` or `process_*` channel variant), and in this case the group or list expansion will be deferred to the specified channel.

For several examples of use of the attribute named by `ldap_reprocess`, see the discussion of [Moderated mailing lists](#).

For aliases defined in Unified Configuration via [alias options](#), see [alias_reprocess](#). Or for aliases defined in legacy configuration in the [alias file](#), see the [\[REPROCESS\] alias file named parameter](#).

39.15.6.51 Direct LDAP attribute name MTA option: `ldap_jettison_domain` (LDAP attribute name list)

(New in Messaging Server 7.3-11.01.) The `ldap_jettison_domain` MTA option specifies the name of an LDAP attribute, by default `mgrpJettisonDomain`, whose value(s) name domains from which to silently **discard** all messages. Glob-style wildcards may be used in the value(s). Multiple attributes and multiple values are allowed.

39.15.6.52 Direct LDAP attribute name MTA option: `ldap_jettison_url` (LDAP attribute name list)

(New in Messaging Server 7.3-11.01.) The `ldap_jettison_url` MTA option specifies the name of an LDAP attribute, by default `mgrpJettisonBroadcasters`, whose value(s) specify a **URL** identifying mail addresses whose messages should be **jettisoned** if sent to this group. Multiple attributes and multiple values are allowed; `mailto:` URLs are acceptable. Each URL is expanded into a list of addresses and each address is checked against the current envelope From address. A match marks the message to be jettisoned and bypasses all other group checks and expansion. **Substitution processing** will be performed on this URL. If bit 6 (value 64) of the `process_substitutions` MTA option is set.

39.15.6.53 Direct LDAP attribute name MTA options: `ldap_list_id` (LDAP attribute name)

RESTRICTED.

The `ldap_list_id` MTA option specifies the name of a group LDAP attribute, by default `mgrpUniqueId`, used to store a unique identifier for the group/list.

The presence of the LDAP attribute named by `ldap_list_id` on group or list entries in LDAP is optional for groups/lists in general, but is required for groups to be managed by the MTA's **MAILSERV**. The attribute's presence and value provides the linkage between `mluser` entries and the group definition in the user/group tree.

39.15.6.54 Direct LDAP attribute name MTA options: `ldap_reject_action` (LDAP attribute name)

The `ldap_reject_action` MTA option specifies the name of a group LDAP attribute, by default `mgrpMsgRejectAction`, whose value will control what to do when a message "fails" a posting check.

For whatever LDAP attribute is named by `ldap_reject_action`, the MTA supports values for the attribute of `reply` and `toModerator`, meaning, respectively, to issue a message rejection, or to redirect the message to the group/list **moderator address**. Note that for a value of `reply` (meaning that the attempted post will be rejected), the value of a `mgrpRejectText` or `mgrpMsgRejectText` LDAP attribute (or more precisely, the value of whatever attributes are named by the `ldap_reject_text` MTA option) may be used to control the error text to send back to the attempting poster.

39.15.6.55 Direct LDAP attribute name MTA options: `ldap_reject_text` (LDAP attribute name list)

The `ldap_reject_text` MTA option specifies the names of group LDAP attributes, by default `mgrpRejectText`, `mgrpMsgRejectText`, whose value will control what error text to use when **rejecting** an attempted posting to a list due to the attempted posting failing an

access check. Because the error text may appear in SMTP responses, it must conform to SMTP response limitations. In particular, it may consist merely of a single line of text limited to the US-ASCII charset. (If the value contains eight bit characters, the entire value will be ignored. If the value contains more than a single line of text, only the first line of text will be used.)

39.15.6.56 Direct LDAP attribute name MTA options:

ldap_auth_policy (LDAP attribute name)

The `ldap_auth_policy` MTA option specifies the name of a group LDAP attribute, by default `mgrpBroadcasterPolicy`. The LDAP attribute named by `ldap_auth_policy` may contain multiple, comma-separated keywords. Supported keywords in the value include `SMTP_AUTH_REQUIRED` and `AUTH_REQ` (require SMTP AUTH use to post), `PASSWORD_REQUIRED` and `PASSWD_REQUIRED` and `PASSWD_REQ` (require a password to post), and `OR` and `AND` (affect combination of multiple access controls), and `NO_REQUIREMENTS`. New in Messaging Server 7.0u4 are the keywords `LIST_OPEN`, `LIST_MEMBERS`, `LIST_MODERATE_NONMEMBERS`, `LIST_MODERATE_MEMBERS`, `LIST_MODERATE`, supported only for lists with a `mgrpUniqueId` (normally MAILSERV lists).

39.15.6.57 Direct LDAP attribute name MTA options:

ldap_cant_url (LDAP attribute name)

The `ldap_cant_url` MTA option names the group LDAP attribute, by default `mgrpDisallowedBroadcaster`, whose value specifies via an [MTA URL](#) a list of (envelope From) addresses *not* allowed to post to the group or list. Note that depending on the setting of the [process_substitutions](#) MTA option, certain substitution sequences may be used in the URL.

Note that the [use_auth_return](#), [use_canonical_return](#), and [use_orig_return](#) MTA options select which form of a message's envelope From address is compared in making the access determination.

39.15.6.58 Direct LDAP attribute name MTA options:

ldap_auth_url (LDAP attribute name)

The `ldap_auth_url` MTA option names the group LDAP attribute, by default `mgrpAllowedBroadcaster`, whose value specifies via an [MTA URL](#) a list of (envelope From) addresses allowed to post to the group or list. Note that depending on the setting of the [process_substitutions](#) MTA option, certain substitution sequences may be used in the URL.

As of JES MS 6.3, when a URL is expanded to get back a list of those sending addresses allowed to post, the MTA will request (for local users) aliases as well as canonical addresses; this means that lists where only members are allowed to post *do* allow local members to post using their defined aliases.

Note that the [use_auth_return](#), [use_canonical_return](#), and [use_orig_return](#) MTA options select which form of a message's envelope From address is compared in making the access determination.

39.15.6.59 Direct LDAP attribute name MTA options:

ldap_cant_domain (LDAP attribute name)

The `ldap_cant_domain` MTA option names the group LDAP attribute, by default `mgrpDisallowedDomain`, whose values specify domains not allowed to post to the group or list.

Note that the [use_auth_return](#), [use_canonical_return](#), and [use_orig_return](#) MTA options select which form of a message's envelope From address is compared in making the access determination.

39.15.6.60 Direct LDAP attribute name MTA options: `ldap_auth_domain` (LDAP attribute name)

The `ldap_auth_domain` MTA option names the group LDAP attribute, by default `mgrpAllowedDomain`, whose values specify domains allowed to post to the group or list. As of JES MS 6.2, the value of the attribute supports use of the asterisk character, `*`, as a wildcard. For instance, a value of `*.domain.com` means to allow all subdomains of `domain.com`, though not `domain.com` itself; to allow `domain.com` and all its subdomains, use two values for the attribute, `domain.com` and `*.domain.com`.

Note that the [use_auth_return](#), [use_canonical_return](#), and [use_orig_return](#) MTA options select which form of a message's envelope From address is compared in making the access determination.

39.15.6.61 Direct LDAP attribute name MTA options: `ldap_maximum_message_size` (LDAP attribute name)

The `ldap_maximum_message_size` MTA option specifies the name of a group LDAP attribute, by default `mgrpMsgMaxSize`, whose value sets an upper limit, in units of [MTA blocks \(block_size\)](#), on how large of message may be posted to the group or list. (Compare with the user LDAP attribute [mailMsgMaxBlocks](#).)

39.15.6.62 Direct LDAP attribute name MTA options: `ldap_maximum_messages_per_day` (LDAP attribute name)

RESTRICTED: Not yet fully implemented.

39.15.6.63 Direct LDAP attribute name MTA options: `ldap_auth_password` (LDAP attribute name list)

The `ldap_auth_password` MTA option specifies the name of a group level LDAP attribute, by default `mgrpAuthPassword`, used to store a password needed to post to the group.

In iMS 5.2 the value of this attribute was saved if the [mgrpBroadcasterPolicy attribute](#) was set to require a password, and the value was then checked against the Approved: field once the header became available. The Approved: field was removed from the header once the check was complete. But this did not allow for routing to the moderator in the event of a password check failure.

In the JES MS 6.0 release and later, the presence of a `mgrpAuthPassword` 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.

Note that the [or_clauses](#) MTA option acts on the `mgrpAuthPassword` attribute in the same way it acts on the other access check attributes.

39.15.6.64 Direct LDAP attribute name MTA options: `ldap_moderator_url` (LDAP attribute name list)

The `ldap_moderator_url` MTA option specifies the name of a group level LDAP attribute, by default `mgrpModerator`, that stores a list of URLs to be expanded into a series of addresses. The interpretation of this address list depends on the value of the LDAP attribute named (by default, `mgrpMsgRejectAction`) by the [ldap_reject_action](#) MTA option.

If the LDAP attribute named by `ldap_reject_action` has its value set to "TOMODERATOR", then the LDAP attribute named by `ldap_moderator_url` specifies the moderator address(es) the message is to be sent to should any of the access checks fail. If the LDAP attribute named by `ldap_reject_action` is missing or has any other value, then the `ldap_moderator_url` attribute's expanded address list is compared with the envelope From address. (Setting particular bits in any of the MTA options [use_auth_return](#), [use_canonical_return](#), or [use_orig_return](#) can control which "form" or version of the envelope From address is compared.) Processing continues if there is a match. If there isn't a match, then the message is again sent to all of the addresses specified by the `ldap_moderator_url` attribute's value(s).

Expansion of the `ldap_moderator_url` attribute's values is implemented by making the values of this attribute the list of URLs for the group; that is, any list of member [RFC 822](#) addresses or DN's previously determined for the group is (temporarily) cleared, and the [delivery options](#) for the group (the `ldap_moderator_url` expanded values) are set to "members". Any additional, subsequent group LDAP attributes (as listed in [Table of MTA LDAP attribute name options](#)) will be ignored at this stage of processing.

Note that as of JES MS 6.3, [substitution processing](#) will be performed on the URL value(s) of the LDAP attribute named by `ldap_moderator_url` if bit 2 (value 4) of the [process_substitutions](#) MTA option is set.

39.15.6.65 Direct LDAP attribute name MTA options: `ldap_group_last_access_time` (LDAP attribute name)

(New in 8.0.) The `ldap_group_last_access_time` MTA option specifies the name of an LDAP attribute used to keep track of the last access time for email groups defined in LDAP. If this attribute is present in a group's LDAP entry, then the MTA will update the attribute each time the group is successfully accessed for purposes of sending mail or expanding a mailing list. [RFC 3339](#) format, an Internet profile of [ISO 8601 format](#), is used, e.g., "2013-09-29T17:38:52Z".

In order to prevent excessive LDAP writes, the LDAP attribute named by the `ldap_group_last_access_time` MTA option is read prior to writing and a write is only done if the current time exceeds the stored time by at least 30 minutes. (A write is also done if the attribute does not contain a valid [RFC 3339](#) time, making it possible to set the initial value to something like "<never accessed>".)

39.15.6.66 Direct LDAP attribute name MTA options: `ldap_group_urll` (LDAP attribute name)

The `ldap_group_url1` MTA option names an LDAP attribute which may be placed on a group entry in LDAP as one way of [identifying members of the group](#). The default LDAP attribute name is `mgrpDeliverTo`; its value should be a [URL](#) identifying the members of the group. See also the `ldap_group_url2` MTA option which names another LDAP attribute (by default `memberURL`) of analogous purpose.

See also the `ldap_url_result_mapping` MTA option; with it, a mapping table can be used to manipulate the value(s) and results of expanding the value(s) of the attributes named by `ldap_group_url1` and `ldap_group_url2`.

39.15.6.67 Direct LDAP attribute name MTA options: `ldap_group_url2` (LDAP attribute name)

The `ldap_group_url2` MTA option names an LDAP attribute which may be placed on a group entry in LDAP as one way of [identifying members of the group](#). The default LDAP attribute name is `memberURL`; its value should be a [URL](#) identifying the members of the group. See also the `ldap_group_url1` MTA option which names another LDAP attribute (by default `mgrpDeliverTo`) of analogous purpose.

See also the `ldap_url_result_mapping` MTA option; with it, a mapping table can be used to manipulate the value(s) and results of expanding the value(s) of the attributes named by `ldap_group_url1` and `ldap_group_url2`.

39.15.6.68 Direct LDAP attribute name MTA options: `ldap_group_dn` (LDAP attribute name)

The `ldap_group_dn` MTA option names an LDAP attribute which may be placed on a group entry in LDAP as one way of [identifying members of the group](#). The default LDAP attribute name is `uniqueMember`.

Typically, and in default use, the value in the LDAP attribute named by `ldap_group_dn` is a DN (Distinguished Name), so that the members of the group are being identified via their position (DN) in the DIT. Note that such a DN may specify an entire subtree of the DIT. Specifically, in basic use the value of the LDAP attribute named by `ldap_group_dn` is substituted into the URL template defined by the [group_dn_template](#) MTA option, whose default value is:

```
ldap:/// $A?mail?sub?(mail=*)
```

where the `uniqueMember` (or whatever LDAP attribute named by `ldap_group_dn`) value is substituted in place of the `$A`.

However, the exact use/interpretation of the LDAP attribute named by `ldap_group_dn` is controlled not only by the [group_dn_template](#) MTA option, but optionally may be further modified via the [GROUP_TEMPLATES](#) mapping table, as well as any special mapping table specified for that group via the group LDAP attribute named by the `ldap_url_result_mapping` MTA option. As of Messaging Server 7.0.5, if a [GROUP_TEMPLATES](#) mapping table exists, then it is used to determine what template (what alternative to `group_dn_template`) to apply to the value of the LDAP attribute named by `ldap_group_dn`. The `GROUP_TEMPLATES` mapping probe is of the form

object-classes | attribute-name | attribute-value

If the mapping sets the `Y` output flag, then the mapping result is used as the template instead of the `group_dn_template` MTA option's value.

Multiple values of the LDAP attribute named by `ldap_group_dn` are allowed on a group entry, *e.g.*, multiple `uniqueMember` values are allowed, but only one attribute name may be specified as `ldap_group_dn`. To use another named LDAP attribute (perhaps in a slightly different way) use `ldap_group_dn2`.

As of Messaging Server 7.0.5 any mapping specified by the [ldap_url_result_mapping MTA option](#) will also be applied to the results produced by the `ldap_group_dn` and `ldap_group_dn2` attributes.

39.15.6.69 ldap_group_dn2 Option

The `ldap_group_dn2` MTA option names an LDAP attribute in which to store a list of DN's, *or other identifiers*, for group members.

The `ldap_group_dn2` MTA option names an LDAP attribute which may be placed on a group entry in LDAP as a way of identifying members of the group. The `ldap_group_dn2` MTA option has no default. The purpose of the `ldap_group_dn2` MTA option, and whatever LDAP attribute it names, is primarily to allow [alternate approaches to identifying group members](#), beyond the identification-via-DN typically achieved via the `uniqueMember` LDAP attribute (more precisely, whatever LDAP attribute is named by the [ldap_group_dn MTA option](#)) being expanded via the URL template specified via the [group_dn_template MTA option](#).

While the LDAP attribute named by `ldap_group_dn2` may be used to store a DN (like the default use of the LDAP attribute named by [ldap_group_dn](#)), more typically it would be used to store some other means of identifying members of a group, with the [GROUP_TEMPLATES mapping table](#) then being configured to make "appropriate" use of the value of the LDAP attribute named by `ldap_group_dn2`. For instance, if

```
ldap_group_dn2=listID
```

and a `GROUP_TEMPLATES` mapping table is configured as

```
GROUP_TEMPLATES
```

```
! Normal use of ldap_group_dn attribute uniqueMember
*|uniqueMember|* $Yldap:/// $A?mail?sub?(mail=*)
! Find users who have a memberOf attribute set to the value of the group's
! memberID attribute
*|listID|* $Yldap:///baseDN-of-users??sub?(memberOf=$A)
```

then "traditional" groups with membership defined via values of the `uniqueMember` LDAP attribute will continue to work as always, while also allowing groups to have membership defined as "all users who have a `memberOf` attribute value matching the group's `listID` attribute value".

Multiple values of the LDAP attribute named by `ldap_group_dn2` are allowed on a group entry, *e.g.*, continuing the example above multiple `memberID` values would be allowed, but

only one attribute name may be specified as `ldap_group_dn2`. Allowing the capability to have two differently named LDAP attributes, potentially expanded via different URL templates, is the reason why `ldap_group_dn2` exists in addition to `ldap_group_dn`.

As of Messaging Server 7.0.5 any mapping specified by the `ldap_url_result_mapping` MTA option will also be applied to the results produced by the `ldap_group_dn` and `ldap_group_dn2` attributes.

39.15.6.70 Direct LDAP attribute name MTA options: `ldap_group_rfc822` (LDAP attribute name list)

The `ldap_group_rfc822` MTA option specifies the name of one or more group LDAP attributes, by default `mgrpRFC822MailMember` and `RFC822MailMember`, whose value(s) will be [RFC 822](#) email addresses of members of the group.

39.15.6.71 Direct LDAP attribute name MTA options: `ldap_url_result_mapping` (LDAP attribute name)

The `ldap_url_result_mapping` MTA option names an LDAP attribute which may be placed on a group entry in LDAP for purposes of manipulating the results of lookups of the `mgrpDeliverTo` and `memberURL` attributes (or more precisely, the attributes named by the `ldap_group_url1` and `ldap_group_url2` MTA options), and as of Messaging Server 7.0-5, also `uniqueMember` (the attributes named by the `ldap_group_dn` and `ldap_group_dn2` MTA options). Currently there is no default value (default attribute) for `ldap_url_result_mapping`; to use this feature, you must choose a (new) attribute and add it to the schema (or disable schema checking).

This attribute's value should name an [MTA mapping table](#) to be applied to any result returned by expanding either a `mgrpDeliverTo` or a `memberURL` attribute, or as of Messaging Server 7.0-5 also a `uniqueMember` attribute as well as any custom attribute named by `ldap_group_dn2`. The mapping probe will be of the form:

```
LDAP-URL | LDAP-result
```

where `LDAP-URL` is the (literal string) value of the `mgrpDeliverTo`, `memberURL`, or `uniqueMember` attribute itself (or custom attribute named by `ldap_group_dn2`), and `LDAP-result` is the value returned from LDAP for that `LDAP-URL` query. If the mapping returns with `$Y` set, then the mapping result string will replace the LDAP result for alias processing purposes. If the mapping returns with `$N` set, then the result will be skipped.

In particular, this mechanism can be used to define groups based on attributes that don't contain proper email addresses.

39.15.6.72 Direct LDAP attribute name MTA options: `ldap_errors_to` (LDAP attribute name)

The `ldap_errors_to` MTA option specifies the name of a group LDAP attribute used to specify an override Envelope From address. Presence of the specified attribute on a group entry is [the critical distinction](#) between whether a group entry is merely a group -- an e-mail auto-forwarder (equivalent to having lots of aliases) -- *vs.* whether the entry defines a true [mailing list](#). In particular, the presence of such an attribute on a group LDAP entry has implications for [notification messages](#) regarding the list definition (*e.g.*, syntactic errors in list

member addresses, or syntactic errors in a list-specific [Sieve filter](#)) or regarding delivery of messages to list members, as well as for the handling of delivery receipt requests.

Typically, the value stored in the specified LDAP attribute will be some normal email address. But two (three, as of JES MS 6.3) special syntaxes are also supported.

Setting the value stored in the specified LDAP attribute to an address of the form `user+*@domain` has a special meaning. The asterisk character will be expanded into a representation of the recipient address; thus a separate copy of the list message is generated for each recipient, with each copy including the intended recipient address as a [subaddress](#) within the return address. If delivery errors subsequently occur, the subaddress will indicate which was the failing address. In some cases, when dealing with remote MTAs that generate nonstandard, uninformative delivery error messages, this can in theory be useful as a way of determining which recipient address(es) failed, even when the bounce message's inner content is relatively uninformative. And it may make processing of such bounce messages by an automated program more convenient. However, the tradeoff is that such per-user-specific return address values require that a separate message copy be generated and sent for each recipient; for a "large" list, with many recipients in the same destination domains, this can be a large increase in overhead (a large decrease in efficiency). And with more prevalent use nowadays of standard format [notification messages](#), the "need" for this sort of approach, with its extra (potentially large) overhead, is much less (since the intended recipient information can instead be extracted from the standard field in the contents of a standard format notification message).

(New in JES MS 6.3, but not working until fix for CR # 6530591.) Setting the value stored in the specified LDAP attribute to the forward slash character, `/`, has a special meaning. It tells the MTA to revert to using the original envelope From address that had been present on the incoming message, yet in all other respects use mailing list semantics. This can be useful for setting up mailing lists that report all forms of list errors to the original sender.

New in JES MS 6.3, the [process_substitutions](#) MTA option can enable use of the `$S` (recipient's subaddress) substitution in the value. This would tend to be of interest when defining a "meta-list".

For users defined via [alias options](#), see instead the [alias_envelope_from](#) alias option; or in legacy configuration [alias file](#) or [alias database](#) definitions, see the [\[ERRORS_TO\]](#) alias file named parameter.

39.15.6.73 Direct LDAP attribute name MTA options: ldap_delay_notifications (LDAP attribute name)

The `ldap_delay_notifications` MTA option specifies the name of a group LDAP attribute, by default `mgrpDelayNotifications`, whose value controls whether `NOTIFY=DELAY` should be set on list messages. In the specified LDAP attribute, the MTA supports values of *yes* and *no*.

39.15.6.74 Direct LDAP attribute name MTA options: ldap_digest_interval (LDAP attribute name)

RESTRICTED. Not yet fully implemented.

39.15.6.75 Direct LDAP attribute name MTA options: ldap_add_header (LDAP attribute name)

The `ldap_add_header` MTA option specifies the name of the group LDAP attribute, by default `mgrpAddHeader`, which contains header lines to add to messages posted to the list. Such header lines might include, for instance, the `List-*`: header lines suggested in [RFC 2369 \(URLs for Mail List Commands through Message Headers\)](#).

See also the `ldap_remove_header` MTA option, for removing header lines from postings to the group or list.

For groups or lists defined via [alias file](#) or [alias database](#) (legacy configuration) or [alias options](#) (Unified Configuration), see instead the [\[HEADER_ADDITION\] alias file named parameter](#) or [alias_header_addition](#) alias option, respectively.

Use of `mgrpAddHeader` (or whatever LDAP attribute is named by `ldap_add_header`) is a very simple approach when unconditional additional of certain header lines is desired. For more complex requirements, consider setting a Sieve filter on the group or list (see the `ldap_filter` MTA option) that makes use of the [Sieve editheader extension](#).

39.15.6.76 Direct LDAP attribute name MTA options: `ldap_remove_header` (LDAP attribute name)

The `ldap_remove_header` MTA option specifies the name of a group LDAP attribute, by default `mgrpRemoveHeader`, whose value specifies header lines to remove from postings to the group or list.

See also the `ldap_add_header` MTA option, for adding header lines to postings to the group or list.

For groups or lists defined via [alias file](#) or [alias database](#) (legacy configuration) or [alias options](#) (Unified Configuration), see instead the [\[HEADER_TRIM\] alias file named parameter](#) or [alias_header_trim](#) alias option, respectively.

Use of `mgrpRemoveHeader` (or whatever LDAP attribute is named by `ldap_remove_header`) is a very simple approach when unconditional removal of certain header lines is desired. For more complex requirements, consider setting a Sieve filter on the group or list (see the `ldap_filter` MTA option) that makes use of the [Sieve editheader extension](#).

39.15.6.77 Direct LDAP attribute name MTA options: `ldap_add_tag` (LDAP attribute name)

The `ldap_add_tag` MTA option specifies the name of a group LDAP attribute used to specify prefix text to insert on the Subject: header line of messages to this recipient/list, analogous to the [alias_tag](#) alias option, the [\[TAG\] named mailing list parameter](#), or the effect of the [Sieve addtag action](#).

As of JES MS 6.3, the vertical bar, `|`, character should not be used in the tag text; in previous versions, the space character should not have been used in tag text, as such use would interfere with the MTA's internal mechanisms for checking whether a tag was already present. As of Messaging Server 7.3, the MTA supports language-tagged values for the attribute named by `ldap_add_tag`, and will select amongst such according to the user's `preferredLanguage` value (more precisely, the value of the LDAP attribute named by the `ldap_preferred_langauge` MTA option).

39.15.6.78 Direct LDAP attribute name MTA options: `ldap_prefix_text` (LDAP attribute name)

The `ldap_prefix_text` MTA option specifies the name of a group LDAP attribute, by default `mgrpMsgPrefixText`, to be used to store text to be inserted into messages posted to the group or list.

This prefix text from the LDAP attribute named by `ldap_prefix_text` (by default the `mgrpMsgPrefixText` attribute) is inserted into messages as they undergo group expansion. Prior to Messaging Server 7.0-3.01, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.0-3.01, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The attribute values are stored in LDAP in UTF-8; this is then converted by the MTA to match the charset of the part that the text is inserted into.

For users defined via alias options, see instead the [alias_prefix_text](#) alias option; or in legacy configuration [alias file](#) or [alias database](#) definitions, see the [\[PREFIX_TEXT\]](#) alias file named parameter. Also see the [addprefix Sieve action](#).

39.15.6.79 Direct LDAP attribute name MTA options: `ldap_suffix_text` (LDAP attribute name)

The `ldap_suffix_text` MTA option specifies the name of a group LDAP attribute, by default `mgrpMsgSuffixText`, to be used to store text to be inserted into messages posted to the group or list.

This suffix text from the LDAP attribute named by `mgrpMsgSuffixText` is inserted into messages as they undergo group expansion. Prior to Messaging Server 7.0-3.01, text could only be inserted into initial, TEXT/PLAIN parts; new in Messaging Server 7.0-3.01, text can be inserted into the first text part within a nested multipart (excluding multipart/alternative). The attribute values are stored in LDAP in UTF-8; this is then converted by the MTA to match the charset of the part that the text is inserted into.

For users defined via alias options, see instead the [alias_suffix_text](#) alias option; or in legacy configuration [alias file](#) or [alias database](#) definitions, see the [\[SUFFIX_TEXT\]](#) alias file named parameter. Also see the [addsufffix Sieve action](#).

39.15.6.80 Direct LDAP attribute name MTA options: `ldap_expandable` (LDAP attribute name)

The `ldap_expandable` MTA option specifies the names of user and group LDAP attributes, by default `mgmanMemberVisibility` and `expandable`, used to define who (in addition to the group or list owner) may view the membership or a group or list; in the context of the MTA, this means who will get the group or list expanded in response to the SMTP EXPN command.

Supported values for the attribute(s) named by the `ldap_expandable` MTA option are:

- `anyone` (which means that the group or list has no specific-to-itself restrictions and all SMTP EXPN commands are performed, unless restricted by general channel or MTA configuration restrictions: see for instance the [expndisable channel option](#)),
- `all` or synonymously `true`, (which means that only authenticated users -- hence users who have an account and provide their password---will be able to expand the group or list),

- and none.

Unrecognized values are interpreted as none.

Note that group or list access controls (e.g., use of attributes such as [mgrpAllowedBroadcaster](#), etc., or an [mgrpBroadcasterPolicy](#) setting of SMTP_AUTH_REQUIRED), also impose restrictions on who is allowed to view list membership. All applicable conditions must be met in order for group or list membership to be viewed (expanded)!

39.15.6.81 Direct LDAP attribute name MTA options:

**ldap_auth_mapping1 (LDAP attribute name),
ldap_auth_mapping2 (LDAP attribute name),
ldap_auth_mapping3 (LDAP attribute name),
ldap_auth_mapping4 (LDAP attribute name)**

New in Messaging Server 7.0.5.

The ldap_auth_mappingN MTA options may be used to specify the names of group attributes to include in [GROUP_AUTH mapping table](#) probes. These MTA options have no default.

39.15.6.82 Direct LDAP attribute name MTA options:

ldap_check_header (LDAP attribute name)

New in Messaging Server 7.0.5.

The ldap_check_header MTA option is used to specify the name of a group attribute used to determine whether or not the message header should be checked for duplication of list recipient addresses. The option does not have a default value.

If the specified attribute has a value of "jettison", the list copy for the recipient specified in the header will jettisoned. If the value is "discard", the system will behave as if the [system Sieve had performed a discard](#) on the list copy for that recipient.

The same capability is also available to aliases defined in the alias file or database via the (also new in Messaging Server 7.0.5) [\[HEADER_CHECK\] named parameter](#), or in Unified Configuration via the [alias_header_check alias option](#).

This capability depends on address typing being enabled - either the [addrtypescan or addrtypescanbccdefault channel option](#) needs to be set on the source channel.

Although this capability has the potential to reduce unwanted message duplicates, *extreme* care should be exercised when it is used. Headers fields are trivially forged, making it possible to send a message that claims to have been sent to someone when in fact it has not. This could potentially be used to suppress the list copy for a given recipient.

39.15.6.83 Head of household LDAP attribute MTA options:

ldap_hoh_filter (LDAP attribute name), ldap_hoh_owner (LDAP attribute name)

The `ldap_hoh_filter` and `ldap_hoh_owner` MTA options specify the names of the LDAP attributes used to store the critical Head of Household data in users who are themselves a Head of Household. These options default to, respectively, `mailSieveRuleSource` and `mail`. That is, `ldap_hoh_filter` specifies the name of the LDAP attribute in which a Head of Household user stores the Sieve filter used for Head of Household purposes (which may or may not be a different Sieve filter than the user's own, personal Sieve filter; the default `mailSieveRuleSource` value causes the Head of Household Sieve filter to be the same as the user's personal Sieve filter, but sites that wish a distinction may set `ldap_hoh_filter` to point to a different, site-specific LDAP attribute). Since proper evaluation of (and especially error reporting regarding) a Sieve filter requires an "owner" e-mail address associated with that Sieve filter, the `ldap_hoh_owner` MTA option specifies what LDAP attribute in the Head of Household user's entry will be the address associated with the Sieve; again, the default value of `mail` means that the Head of Household user's own, personal e-mail address will be used, but sites that wish a distinction may set `ldap_hoh_owner` to some different, site-specific LDAP attribute.

These MTA options specify the names of the LDAP attributes to return when a user entry has parental controls/head of household controls set on it (see the [ldap_parental_controls](#) MTA option) so that a lookup of the user's "parent" (see the [ldap_filter_reference](#) MTA option) is performed: in the "parent" entry, the attributes specified by `ldap_hoh_filter` and `ldap_hoh_owner` are found and their values returned.

39.15.6.84 Direct LDAP attribute name MTA options:

`ldap_attr_domain1_schema2` (LDAP attribute name)

The `ldap_attr_domain1_schema2` MTA option may be used to override, for MTA domain lookup purposes, the Domain Map library code's default use of the `sunPreferredDomain` LDAP attribute as the name of the Schema 2 mode domain level LDAP attribute which specifies the domain name within the domain entry. If this MTA option is not set (the default), then the Domain Map's default of `sunPreferredDomain` is used, as normal.

39.15.6.85 Direct LDAP attribute name MTA options:

`ldap_attr_domain2_schema2` (LDAP attribute name)

The `ldap_attr_domain2_schema2` MTA option may be used to override, for MTA domain lookup purposes, the Domain Map library code's default use of the `associatedDomain` LDAP attribute as the name of the Schema 2 mode domain level LDAP attribute which specifies any secondary domain names (aliases for the canonical domain name) within the domain entry. If this MTA option is not set (the default), then the Domain Map's default of `associatedDomain` is used, as normal.

39.15.6.86 Direct LDAP attribute name MTA options:

`ldap_attr_domain_search_filter` (LDAP attribute name)

The `ldap_attr_domain_search_filter` MTA option specifies the name of the LDAP attribute in the global configuration template area (see the [ldap_global_config_templates](#) MTA option) that is used to store the domain search filter template. For instance, one attribute that might be used for such a purpose (hence to which this option might be set) would be `inetDomainSearchFilter`.

39.15.6.87 Direct LDAP attribute name MTA options:

`ldap_domain_attr_basedn` (LDAP attribute name)

The `ldap_domain_attr_basedn` MTA (and `base`) option names the domain LDAP attribute, by default `inetDomainBaseDn`, used to store the base DN for the domain's users and groups.

The presence in a domain entry of the attribute named by `ldap_domain_attr_basedn` is not always obligatory with Schema 2, as with Schema 2 in the domain attribute's absence user and group entries will be assumed to reside directly under the domain entry.

Note that the mapping table domain map attribute substitution `}${domain,_base_dn}` returns either the value of the LDAP attribute named by the `ldap_domain_attr_basedn` MTA option (so normally the value of the `inetDomainBaseDN` LDAP attribute), or if no such LDAP attribute is set as can be the case in Schema 2 mode, returns a constructed DN corresponding to the DN for the domain entry.

39.15.6.88 Direct LDAP attribute name MTA options: `ldap_domain_attr_alias` (LDAP attribute name)

The `ldap_domain_attr_alias` MTA (and `base`) option specifies the name of an LDAP attribute (by default `aliasedObjectName`) used to identify domain alias entries in the directory. The attribute is present only on a domain alias entry, not on the canonical domain entry; it contains the DN of the entry for which it is an alias. It is used only in Schema 1 or in Schema 2 compatibility mode (with a DC Tree), not in Schema 2 native mode (no DC Tree).

39.15.6.89 Direct LDAP attribute name MTA options: `ldap_domain_attr_uplevel` (LDAP attribute name)

The `ldap_domain_attr_uplevel` MTA option specifies the name of a domain-level LDAP attribute used to store a domain-specific analogue of the `domain_uplevel` MTA option that overrides that MTA option for this specific domain. That is, the attribute named by this option stores a bitmask value controlling certain aspects of domain name searching and usage. Currently only bits 0 and 2 (values 1 and 4) are used from this value; the other bits of the general `domain_uplevel` MTA option remain in effect.

Note that the attribute named by the `ldap_domain_attr_uplevel` MTA option is only consulted if the domain is looked up. This means that setting bit 0 of this value to 1 for a domain won't make subdomains of the domain match unless bit 0 of `domain_uplevel` is also set. As such, the way to get subdomain matching for some domains but not others is to set bit 0 of `domain_uplevel` (thus enabling subdomain matches for all domains), and then clear bit 0 of the `ldap_domain_attr_uplevel` attribute for the domains where you don't want uplevel matching to occur.

39.15.6.90 `ldap_domain_attr_mailserv` Option

The `ldap_domain_attr_mailserv` MTA option specifies the name of a domain level LDAP attribute. It has no default, but the recommended LDAP attribute name to use is `inetDomainMailserv`.

39.15.6.91 Direct LDAP attribute name MTA options: `ldap_domain_attr_canonical` (LDAP attribute name)

The `ldap_domain_attr_canonical` MTA option names the domain LDAP attribute, by default `inetCanonicalDomainName`, used to store the canonical domain name.

In Schema 1 mode, any domain alias entry needs such an attribute pointing back to the "real" (canonical) domain name. And in either Schema, any cases of multiple actual (non-alias) domain entries with "overlapping" users require use of such an attribute.

Note that the mapping table domain map attribute substitution `$_domain,_canonical_name_` returns either the value of the LDAP attribute named by the `ldap_domain_attr_canonical` MTA option (so normally the value of the `inetCanonicalDomainName` LDAP attribute), or if no such LDAP attribute is set, returns the domain name.

39.15.6.92 Direct LDAP attribute name MTA options:

`ldap_domain_attr_uid_separator` (LDAP attribute name)

The `ldap_domain_attr_uid_separator` MTA (and `base`) option names the domain LDAP attribute, by default `domainUidSeparator`, used to store what the separator character is between UIDs and domains for addresses in this domain. This option is used both by the MTA, and by the authentication code; the authentication code looks first for the option to be set at base level, but if not set there, the authentication code will use the MTA level option setting.

39.15.6.93 Direct LDAP attribute name MTA options:

`ldap_domain_attr_subaddress` (LDAP attribute name)

The `ldap_domain_attr_subaddress` MTA option specifies the name of the attribute that controls whether or not the domain supports subaddressing. Subaddress handling will be disabled if the specified attribute has a value of "No", "0", or "false". This option has no default.

39.15.6.94 Direct LDAP attribute name MTA options:

`ldap_domain_attr_routing_hosts` (LDAP attribute name)

The `ldap_domain_attr_routing_hosts` MTA option names the domain LDAP attribute, by default `mailRoutingHosts`, used to specify the hosts that are responsible for performing routing for this domain. If this MTA is one such host, then the user address will be looked up and attributes processed. Otherwise, the address will be routed onwards: by default, just routing based on rewriting the address, but if the MTA option `route_to_routing_host=1` is set, then the first `mailRoutingHosts` value will be inserted into the address as a source route (hence the rewriting routing will depend upon that host name).

Note that delivery options can be marked as mail host independent, thereby meaning that processing should occur regardless of whether this MTA is one of the `mailRoutingHosts`; see the `delivery_options` MTA option.

39.15.6.95 Direct LDAP attribute name MTA options:

`ldap_domain_attr_smarthost` (LDAP attribute name)

The `ldap_domain_attr_smarthost` MTA option names the domain LDAP attribute, by default `mailRoutingSmartHost`, used for routing of addresses in the domain but without a directory entry. That is, if a user address is not found in the directory, then route onwards, inserting the `mailRoutingSmartHost` value into the address as a source route.

Since the value of the `mailRoutingSmartHost` attribute (or whatever attribute is named by `ldap_domain_attr_smarthost`) will be used as a source route, note that that means that the actual routing of the address will depend on how the MTA has been configured to route such an address. In particular, best practice is to specify a FQDN (Fully Qualified Domain Name) as the value, since comprehensive configuration for proper routing of such domain names is part of normal MTA configuration. Use of less recommended substitutes, such as short-form hostnames or IP literal addresses, will only yield desired results if the MTA has been explicitly configured to route such substitute names appropriately.

39.15.6.96 Direct LDAP attribute name MTA options: `ldap_domain_attr_status` (LDAP attribute name)

The `ldap_domain_attr_status` MTA (and `base`) option names the domain LDAP attribute, by default `inetDomainStatus`, whose value specifies the current status of the domain. (The analogous user level attribute is `inetUserStatus` or whatever user LDAP attribute is named by the `ldap_user_status` MTA option; an analogous group attribute can be defined via the `ldap_group_status` MTA option. Compare also with the `ldap_domain_attr_mail_status` MTA option naming the domain LDAP attribute specifying the current mail status of the domain.) Valid values for the attribute named by the `ldap_domain_attr_status` option are `active`, `inactive`, or `deleted`. If no such attribute is present, or is present but with no value, a value of `active` is assumed.

39.15.6.97 Direct LDAP attribute name MTA options: `ldap_domain_attr_mail_status` (LDAP attribute name)

The `ldap_domain_attr_mail_status` MTA (and `base`) option names the domain LDAP attribute, by default `mailDomainStatus`, whose value specifies the current mail status of the domain. (The analogous user level attribute is `mailUserStatus` or whatever user LDAP attribute is named by the `ldap_user_mail_status` MTA option; the analogous group attribute is `inetMailGroupStatus` or whatever group LDAP attribute is named by the `ldap_group_mail_status` MTA option. Compare also with the `ldap_domain_attr_status` MTA option naming the domain LDAP attribute specifying the current general status of the domain.) Valid values for the attribute named by the `ldap_domain_attr_mail_status` option are: `active`, `inactive`, `deleted`, `hold`, `disabled`, `overquota`, and (new in JES MS 6.0) `unused` and `removed` and (new in 8.0) `nonlocal`; other values are interpreted as `inactive`. Note that the `imquotacheck` utility is what updates `mailDomainStatus` to set it to `overquota`.

39.15.6.98 Direct LDAP attribute name MTA options: `ldap_domain_attr_blocklimit` (LDAP attribute name(s))

The `ldap_domain_attr_blocklimit` MTA option specifies the name of an LDAP attribute -- or a list of such names -- used to store a size limit on messages to users in the domain. The default LDAP attribute name is `mailDomainMsgMaxBlocks`.

The effect of the attribute named by `ldap_domain_attr_blocklimit` is a destination (recipient) analogue of the effect of whatever attribute is named by `ldap_domain_attr_sourceblocklimit` MTA option, which limits the size of message that may be sent by users in the domain. The attribute named by `ldap_domain_attr_blocklimit` may also be considered as the domain-level analogue of the user-level `mailMsgMaxBlocks` attribute (or whatever attribute is named by the

`ldap_blocklimit` MTA option) and group `mgrpMsgMaxSize` attribute (or whatever attribute is named by the `ldap_maximum_message_size` MTA option).

New in JES MS 6.3, this attribute is also checked during `reverse_url` lookups, and will be used (for messages that have no return-of-content policy already set) to decide whether the NOTARY non-return-of-content flag should be set.

A new effect in JES MS 6.3-0.15 is that a per-domain setting such as this will override more general settings (and a per-user setting will override even a per-domain setting), rather than (as previously) the minimum of all applicable limits being applied. Thus new in JES MS 6.3-0.15, users in a particular domain maybe allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this domain attribute while a general smaller limit (such as that set via the `blocklimit` channel option) remains in effect.

39.15.6.99 Direct LDAP attribute name MTA options:

`ldap_domain_attr_conversion_tag` (LDAP attribute name(s))

The `ldap_domain_attr_conversion_tag` MTA option specifies the name of a domain LDAP attribute, by default `mailDomainConversionTag`; the value stored in the specified attribute will be applied as [conversion tags](#) for messages sent to users or groups associated with this domain.

39.15.6.100 Direct LDAP attribute name MTA options:

`ldap_domain_attr_source_conversion_tag` (LDAP attribute name(s))

The `ldap_domain_attr_source_conversion_tag` MTA option specifies the name of a domain LDAP attribute; there is no default. The value stored in the specified attribute will be applied as [conversion tags](#) for messages sent from users or groups associated with this domain.

39.15.6.101 Direct LDAP attribute name MTA options:

`ldap_domain_attr_optinN` (LDAP attribute name list)

For values of `N=1--8` (`N=1--4` new in JES MS 6.2-0.01, `N=5--8` new in JES MS 6.3-0.15), the `ldap_domain_attr_optinN` MTA option names a domain LDAP attribute used to store the fact and specific "opt-in" value string that recipients in the domain are "opted-in" to filtering performed by [spam/virus filter package package N](#). `ldap_domain_attr_optin` is a synonym for `ldap_domain_attr_optin1`.

The `ldap_optinN` MTA options provide analogous per-user (rather than per-domain) opt-in capability.

39.15.6.102 Direct LDAP attribute name MTA options:

`ldap_domain_attr_presence` (LDAP attribute name)

RESTRICTED. Not yet used.

39.15.6.103 Direct LDAP attribute name MTA options:

`ldap_domain_attr_autosecretary` (LDAP attribute name)

RESTRICTED. Not yet used.

39.15.6.104 Direct LDAP attribute name MTA options:

ldap_domain_attr_nosolicit (LDAP attribute name)

The `ldap_domain_attr_nosolicit` MTA option specifies the name of a domain-level LDAP attribute used to store solicitation labels to block for recipients in the domain. The attribute named is thus a domain-level analogue of the user-level attribute named by the `ldap_nosolicit` MTA option; and supplements any channel-level solicitation blocking configured via `destinationnosolicit` and `sourcenosolicit` channel options.

See [RFC 3865 \(No Soliciting SMTP Extension\)](#) for background on the NO-SOLICITING SMTP extension.

39.15.6.105 Direct LDAP attribute name MTA options:

ldap_domain_attr_autoreply_timeout (LDAP attribute name)

The `ldap_domain_attr_autoreply_timeout` MTA option specifies the name of an LDAP attribute which is a domain-level analogue of the user-level LDAP attribute named by the `ldap_autoreply_timeout` MTA option. That is, the LDAP attribute named by `ldap_domain_attr_autoreply_timeout` stores the duration, in hours, for successive vacation (autoreply) responses to any given mail sender, to be used for vacation messages generated on behalf of users in this domain who do not have their own, user-level, specific timeout set (no `ldap_autoreply_timeout` attribute set).

`ldap_domain_attr_autoreply_timeout` attribute's value will not be used only when a user has `mailAutoReplyMode=echo`. If the attribute's 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. If neither this domain-level attribute is set, nor the user-level attribute is set, then the timeout used will be that set via the `autoreply_timeout_default` MTA option.

39.15.6.106 Direct LDAP attribute name MTA options:

ldap_domain_attr_default_mailhost (LDAP attribute name)

RESTRICTED: Not supported by the MMP.

The `ldap_domain_attr_default_mailhost` MTA option specifies the name for a domain LDAP attribute, (no default but the `preferredMailHost` LDAP attribute formerly used in provisioning would be one possibly appropriate attribute to also use for this purpose), used to store a default mail host to be used if a user or group entry in the domain has no explicit `mailHost` value (more precisely, no value for the LDAP attribute named by the `ldap_mailhost` MTA option).

39.15.6.107 Direct LDAP attribute name MTA options:

ldap_domain_attr_disk_quota (LDAP attribute name(s))

RESTRICTED. Not yet fully implemented.

39.15.6.108 Direct LDAP attribute name MTA options:

ldap_domain_attr_message_quota (LDAP attribute name(s))

RESTRICTED. Not yet fully implemented.

See the user level LDAP attribute named by the [ldap_message_quota](#) MTA option. Also see the Message Store option [defaultmessagequota](#).

39.15.6.109 Direct LDAP attribute name MTA options:

ldap_domain_attr_filter (LDAP attribute name(s))

The `ldap_domain_attr_filter` MTA option specifies the name of an LDAP attribute (by default `mailDomainSieveRuleSource`) -- or a list of such names -- used to store a specific-to-that-domain [Sieve filter](#).

In operational terms, such a [domain Sieve filter](#) is applied to any recipients in that domain; it is added to any personal Sieve filter for the recipient and applied at the same time as the recipient user's personal Sieve filter.

39.15.6.110 Direct LDAP attribute name MTA options:

ldap_domain_attr_capture (LDAP attribute name(s))

New in the 8.0 release. The `ldap_domain_attr_capture` MTA option specifies the name of a domain LDAP attribute that will be used to trigger automatic "capturing" of user or group e-mail messages for all users and groups in the domain. There is no default - no pre-defined LDAP attribute for this purpose. Typically, the LDAP attribute defined for this purpose, and named by `ldap_domain_attr_capture`, should be set up with an [ACI so that it is not even visible, let alone modifiable, by users](#).

The value(s) of the LDAP attribute named by `ldap_domain_attr_capture` should be the address(es) to which the "captured" message copies are supposed to be sent. When a domain has this attribute specified, then both messages sent to users in the domain, as well as messages from users in the domain, will also have a "capture" copy (normally an encapsulated copy with an entirely new message envelope) sent to the specified address.

Note that the LDAP attribute(s) specified by the [ldap_capture](#) MTA option have similar semantics except the attribute(s) are placed in the user or group entry instead of at the domain level.

The [capture_format_default](#) MTA option controls whether message copies generated due to use of the LDAP attribute named by `ldap_domain_attr_capture` are generated in DSN encapsulated format, *vs.* another formats such as envelope "journal" format.

39.15.6.111 Direct LDAP attribute name MTA options:

ldap_domain_attr_report_address (LDAP attribute name(s))

The `ldap_domain_attr_report_address` MTA option specifies the name of an LDAP attribute (by default `mailDomainReportAddress`) -- or a list of such names -- used to store the address of the [domain postmaster](#).

The domain postmaster, if set, is used as the header From: address in DSNs reporting problems associated with recipient addresses in the domain. It is also used (in certain cases) when reporting problems to users within the domain regarding errors associated with non-local addresses. If this attribute is not set, then in those certain cases the reporting address

will default to `postmaster@domain`. (This is the specific domain name, as opposed to the default or host domain.) But note that regardless of whether or not this attribute is set, there are a number of other cases where the overall host's [postmaster address](#) will be used, rather than any domain-specific postmaster address.

39.15.6.112 Direct LDAP attribute name MTA options:

`ldap_domain_attr_catchall_address` (LDAP attribute name(s))

The `ldap_domain_attr_catchall_address` MTA option specifies the name of an LDAP attribute (by default `mailDomainCatchallAddress`) -- or a list of such names -- used to store a "catch all" address for the domain: an address to which to route any messages to recipients apparently in the domain but with an unrecognized local-part (*unknown-user@domain-name*).

39.15.6.113 Direct LDAP attribute name MTA options:

`ldap_domain_attr_catchall_mapping` (LDAP attribute name(s))

The `ldap_domain_attr_catchall_mapping` MTA option specifies the name of an LDAP attribute (by default `mailDomainCatchallMapping`) -- or a list of such names -- used to store the name of an [MTA mapping table](#).

A mapping table named in such an attribute will be consulted when an address associated with the domain fails to match any particular user entry. The format of the mapping table probe is the same as that of the [FORWARD mapping table](#), and is affected by any setting of the [use_forward_database](#) MTA option in the same way as the FORWARD mapping table probe is affected. The effect of the mapping is that if the mapping sets the `$Y` metacharacter, then the resulting string will replace the address being processed.

New in Messaging Server 7.0-0.04, the use/non-use of POP-before-SMTP can be checked in a domain catchall mapping by checking for presence or absence of the `$P` input flag; that is, by checking for `$:P` or `$;P` respectively.

39.15.6.114 Direct LDAP attribute name MTA options:

`ldap_domain_attr_sourceblocklimit` (LDAP attribute name(s))

The `ldap_domain_attr_sourceblocklimit` MTA option specifies the name of an LDAP attribute -- or a list of such names -- used to store a size limit on messages sent by users in the domain.

The effect of the attribute named by `ldap_domain_attr_sourceblocklimit` is a source (sending) analogue of the effect of the `mailDomainMsgMaxBlocks` attribute (or whatever attribute is named by `ldap_domain_attr_blocklimit`), which limits the size of message that may be received by users in the domain. The attribute named by `ldap_domain_attr_sourceblocklimit` may also be considered as the domain-level analogue of the user-level attribute named by `ldap_sourceblocklimit`.

A new effect in JES MS 6.3-0.15 is that a per-domain setting such as this will override more general settings (and a per-user setting will override even a per-domain setting), rather than (as previously) the minimum of all applicable limits being applied. Thus new in JES MS 6.3-0.15, users in a particular domain maybe allowed to send large messages as an exception to more general, smaller limits, by setting a large value for this domain attribute while a general smaller limit (such as that set via the [sourceblocklimit](#) channel option) remains in effect.

39.15.6.115 Direct LDAP attribute name MTA options:

`ldap_domain_attr_source_channel` (LDAP attribute name)

The `ldap_domain_attr_source_channel` MTA option specifies the name of an LDAP attribute used, when the `userswitchchannel` channel option is in effect, to store the name of an MTA channel to consider as the source channel for messages submitted from users in this domain. See also the `ldap_source_channel` MTA option which names an analogous user-level LDAP attribute. If both a user-level and domain-level attribute are set, the user-level value overrides the domain-level value.

39.15.6.116 Direct LDAP attribute name MTA options:

`ldap_domain_attr_recipientlimit` (LDAP attribute name)

The `ldap_domain_attr_recipientlimit` MTA option specifies the name of a per-domain LDAP attribute, analogous to the per-user `ldap_recipientlimit` MTA option, the `recipientlimit` channel option, and the per-SMTP-server `ALLOW_RECIPIENTS_PER_TRANSACTION` TCP/IP-channel-specific option.

Compare with the `ldap_domain_attr_recipientcutoff` MTA option which specifies the name of a per-domain LDAP attribute for controlling a related, but not identical, effect.

New behavior in JES MS 6.3 is that a per-domain setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular domain can be allowed to send messages to many recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.

39.15.6.117 Direct LDAP attribute name MTA options:

`ldap_domain_attr_recipientcutoff` (LDAP attribute name)

The `ldap_domain_attr_recipientcutoff` MTA option specifies the name of a per-domain LDAP attribute, analogous to the per-user LDAP attribute named by the `ldap_recipientcutoff` MTA option, the `recipientcutoff` channel option, and the per-SMTP-server `REJECT_RECIPIENTS_PER_TRANSACTION` TCP/IP-channel-specific option.

Compare with the `ldap_domain_attr_recipientlimit` MTA option which specifies the name of a per-domain LDAP attribute for controlling a related, but not identical, effect.

New behavior in JES MS 6.3 is that a per-domain setting such as this will override more general settings, rather than (as previously) the minimum of all applicable limits being applied; thus new in JES MS 6.3, a particular domain can be allowed to send messages to many recipients as an exception to more general, smaller limits, by setting a large value for this attribute while general small limits remain in effect.

39.15.6.118 Direct LDAP attribute name MTA options:

`ldap_domain_attr_detourhostoptin` (LDAP attribute name)

(New in Messaging Server 7.0.5.) The `ldap_domain_attr_detourhostoptin` MTA option specifies the name of a per-domain LDAP attribute, analogous to the per-user LDAP attribute named by the `ldap_detourhost_optin` MTA option. If this LDAP attribute named by `ldap_domain_attr_detourhostoptin` has the special value (if any) specified by the

[aliasdetourhost_null_optin](#) MTA option, that will be considered equivalent to the domain attribute being absent.

39.15.6.119 Direct LDAP attribute name MTA options: ldap_creation_date (LDAP attribute name)

(New in 8.0.) The `ldap_creation_date` MTA option specifies the name of a user or group LDAP attribute used to store the account creation date for the user or group. The actual date value stored in such an attribute must be in [RFC 3339 format \(a superset of the format used for vacation start and end times\)](#).

39.15.6.120 Direct LDAP attribute name MTA options: ldap_domain_attr_creation_date (LDAP attribute name)

(New in 8.0.) The `ldap_domain_attr_creation_date` MTA option specifies the name of a domain LDAP attribute used to store the creation date for that domain. The actual date value stored in such an attribute must be in [RFC 3339 format \(a superset of the format used for vacation start and end times\)](#).

39.15.7 Direct LDAP attributes returned upon authentication MTA options

By default, the MTA and the authentication library assume a [particular sort of LDAP schema](#). However, as of the 8.0 release the exact attribute names that the authentication library returns (for the MTA to use) are configurable via various `ldap_auth_attr_*` MTA options.

39.15.7.1 LDAP attributes returned upon authentication MTA options: ldap_auth_attr_mail_host (LDAP attribute name)

The `ldap_auth_attr_mail_host` MTA option specifies the name of an LDAP attribute whose value should be returned, upon successful authentication, as the "mail host" for the user. The default such LDAP attribute, if this option is not specified, is `mailHost`.

39.15.7.2 LDAP attributes returned upon authentication MTA options: ldap_auth_attr_sender (LDAP attribute name)

(New in 8.0.) The `ldap_auth_attr_sender` MTA option specifies the name of the LDAP attribute whose value should be returned (upon successful authentication) as the "authenticated sender". The default such LDAP attribute, if this option is not set, is the `mail` attribute.

39.15.7.3 LDAP attributes returned upon authentication MTA options: ldap_auth_attr_submit_channel (LDAP attribute name)

(New in 8.0.) The `ldap_auth_attr_submit_channel` MTA option specifies the name of an LDAP attribute whose value should be returned, upon successful authentication, as the desired source channel name (the source channel to which to "switch" when [saswitchchannel](#) channel option is set). The default LDAP attribute name, if this option is not set, is `mailSMTPSubmitChannel`.

39.15.7.4 LDAP attributes returned upon authentication MTA options: `ldap_auth_attr_recall_secret` (LDAP attribute name)

The `ldap_auth_attr_recall_secret` MTA option specifies the name of the LDAP attribute where a user's general recall secret is stored. This option has no default.

See the [trackinggenerate](#) channel option for further discussion of the purpose and use of the value stored in the specified LDAP attribute.

39.15.8 LDAP lookup cache MTA options

There are a number of MTA options controlling the caching of LDAP and URL lookup results. See also the `ldap_timeout` MTA option, controlling how long the MTA waits for a response from LDAP before timing out a query attempt.

See also the [Sieve filter caching MTA options](#), which control caching of parsed Sieve filters regardless of source (so including but not limited to Sieve filters fetched from LDAP).

Note that the MTA has no specific-to-itself setting for caching of authentication (SMTP AUTH) results from LDAP; that is instead controlled by the general (base level) settings for the `authcachesize` and `authcachettl` options (in legacy configuration, the `configutil` parameters `service.authcachesize` and `service.authcachettl`).

The MMP has its own cache of the results of searching for users in LDAP; see the `ldapcachesize` and `ldapcachettl` options.

The `cache_debug` MTA option enables low-level debugging regarding the MTA's caching of LDAP lookup results.

39.15.8.1 LDAP and URL lookup cache options:

**`alias_entry_cache_negative` (0 or 1),
`alias_entry_cache_size` (integer),
`alias_entry_cache_timeout` (integer)**

The `alias_entry_cache_*` MTA options control some performance tuning relevant when operating in direct LDAP mode, in particular when [alias_urlN options](#) are being used. When `alias_urlN` lookups are performed, the results can be cached; that is, an in memory cache is maintained of the results of "recent" such lookups. The `alias_entry_cache_size` option, which defaults to 1000, controls how many results are cached. The `alias_entry_cache_timeout` option, which defaults to 600, controls how long in seconds to maintain cache results. The `alias_entry_cache_negative` option, which takes a boolean argument and defaults to 0 (false), controls whether or not negative results (that is, failures to find an alias) are cached.

There is a trade-off between performance on the one hand, *vs.* memory usage and speed with which changes to the LDAP entries take effect on the other hand.

39.15.8.2 Domain match cache control

(`domain_match_cache_size`, `domain_match_cache_timeout`)

As of JES MS 6.0, the `domain_match_cache_size` MTA option (as well as the [domain_match_cache_timeout MTA option](#)) is mostly irrelevant, since as of JES MS 6.0

there is an underlying domain map cache of domains. That is, the underlying domain lookup code used by the MTA as well as other Messaging Server components now maintains a (large) cache of lookup results; see the [ldap_domain_timeout](#) MTA option for a discussion of the timeout on the underlying domain lookup cache entries. The MTA's own private-to-the-MTA cache has thus become mostly redundant -- it's a cache in front of a cache (though its entries are smaller in the case of negative matches, so potentially if one was especially concerned about optimizing the case of caching of large numbers of negative matches, the MTA's private cache might be slightly useful).

The `domain_match_cache_*` options control some performance tuning formerly (in iMS 5.2) relevant when operating in direct LDAP mode, in particular when a [rewrite rule with a \\$V](#) in the template is being used. When such [\\$V domain map lookups](#) are performed, the results can be cached by the MTA (apart and in addition to the caching done by the underlying domain map LDAP lookup code); that is, an in memory cache is maintained by the MTA of the results of "recent" such lookups. The `domain_match_cache_size` MTA option, which defaults to 100000, controls how many results are cached. The [domain_match_cache_timeout MTA option](#), which defaults to 600, specifies how long in seconds to maintain cache results. But note that all that's being cached here by the MTA is whether or not the domain is "local", (that is, in the DIT). The actual values of attributes, such as `domainStatus`, is cached in the underlying domain map code; see the [ldap_domain_timeout](#) MTA option for a discussion of the timeout for that underlying cache.

There is a trade-off between performance on the one hand, *vs.* memory usage and speed with which changes to the LDAP entries take effect on the other hand.

See also: [MTA options](#), [Option value syntax in legacy configuration](#), [Getting option changes to take effect](#), [MTA options listed alphabetically](#), [MTA options listed by functional groups](#), [LDAP and URL lookup cache options](#), [Direct LDAP domain lookup options](#), [Direct LDAP MTA options](#), [ldap_domain_timeout MTA option](#), [domain_match_cache_timeout MTA option](#), [Rewrite rule \\$V LDAP lookup](#).

39.15.8.3 LDAP lookup cache MTA options: `ldap_domain_timeout` (integer)

The `ldap_domain_timeout` option (available at both base and MTA levels) controls the retention time (in seconds) for entries in the domain map cache. The default is -900; as the value used is the absolute value of the `ldap_domain_timeout` setting, this corresponds to 15 minutes. If setting `ldap_domain_timeout` explicitly, set it to a positive value so that the MTA can detect that it has indeed been intentionally set.

39.15.8.4 LDAP lookup cache MTA options: `reverse_address_cache_size` (integer) and `reverse_address_cache_timeout` (integer)

The `reverse_address_cache_*` MTA options control some performance tuning relevant when operating in direct LDAP mode, in particular when the [reverse_url](#) MTA option is set. When such `reverse_url` address reversal LDAP lookups are performed, the results can be cached; that is, an in-memory cache is maintained of the results of "recent" such lookups. The `reverse_address_cache_size` option, which defaults to 100000, controls how many results are cached. The `reverse_address_cache_timeout` option, which defaults to 600, specifies how long in seconds to maintain cache results. There is a trade-off between performance on the one hand, *vs.* memory usage and speed with which changes to the LDAP entries take effect on the other hand.

39.15.8.5 LDAP lookup cache MTA options: `url_result_cache_size` (integer) and `url_result_cache_timeout` (integer)

LDAP lookups done from callouts in rewrite rules or mapping tables may be cached; that is to say, `$]` . . . [lookups may be cached. The `url_result_cache_size` MTA option controls the size of this cache; the default is 10000. The `url_result_cache_timeout` MTA option controls the timeout, in seconds, for entries in this cache; the default is 600.

39.16 Directory location MTA options

As of Messaging Server 7.0-0.04, many formerly configurable (via [MTA Tailor option](#)) [MTA directory locations](#) are now hard-coded. `tmpdir` (formerly called `imta_tmp`) and `langdir` (formerly called `imta_lang`) are still configurable.

39.16.1 Directory location MTA options: `tmpdir` (directory path)

The `tmpdir` MTA option (formerly the MTA option was called `imta_tmp`) specifies the temporary file directory; it defaults to `/tmp/` (trailing slash included).

On Linux, this option should instead be set to `/dev/shm/`.

39.16.2 Directory location MTA options: `langdir` (dir- path or list of dir-paths)

The `langdir` MTA option specifies the directory containing localized [MTA return and disposition templates](#). (Formerly, this was called `imta_lang`.)

39.17 DKIM MTA options

New in Messaging Server 7.0.5 are several options that relate to DKIM handling. See also the [dkim*](#) and [destinationdkim*](#) channel options.

39.17.1 DKIM MTA options: `dkim_ignore_domains` (list of domain names)

(New in Messaging Server 7.0.5.) The `dkim_ignore_domains` MTA option modifies the effect of the [dkimpreserve](#), [destinationdkimpreserve](#), [dkimremove](#), and [destinationdkimremove](#) channel options. `dkim_ignore_domains` specifies a list of domain names which will be *ignored* in DKIM-Signature: header lines, both for DKIM signature preservation purposes (that is, *ignored* when either the [dkimpreserve](#) or [destinationdkimpreserve](#) channel options is specified), as well as for DKIM signature removal purposes (that is, *ignored* when either the [dkimremove](#) or [destinationdkimremove](#) channel option is specified). A domain name listed in `dkim_ignore_domains` will be, for the MTA's DKIM handling purposes, invisible on DKIM-Signature: header lines---such a domain will trigger neither [passthrough](#) mode to preserve DKIM signatures, nor DKIM signature removal.

39.17.2 DKIM MTA options: `dkim_preserve_domains` (list of domain names)

(New in Messaging Server 7.0.5.) The `dkim_preserve_domains` MTA option modifies the effect of the `dkimpreserve` and `destinationdkimpreserve` channel options. When such a channel option is being applied, after first checking each domain on a DKIM-Signature: header line against `dkim_ignore_domains` for domain names to ignore, next the MTA will check the list of domain names specified by `dkim_preserve_domains` and if a match is found, `passthrough` mode will be triggered (further general message processing is terminated). Note that once such a match on the `dkim_preserve_domains` list is found, the MTA need not, and does not, bother to scan further in the message or list for additional matches.

39.17.3 DKIM MTA options: `dkim_remove_domains` (list of domain names)

(New in Messaging Server 7.0.5.) The `dkim_remove_domains` MTA option modifies the effect of the `dkimremove` and `destinationdkimremove` channel options. When such a channel option is being applied, after first checking each domain on a DKIM-Signature: header line against `dkim_ignore_domains` for domain names to ignore, next the MTA will check the list of domain names specified by `dkim_remove_domains` and if a match is found, then the corresponding DKIM-Signature: field is removed from the message.

39.18 DNS lookup MTA options

In addition to the `blocked_mail_from_ips` and `return_envelope` MTA options with some DNS lookup effects, see also [TCP/IP channels](#) for additional, but channel-specific, DNS-related options. And see [SPF MTA options](#) for DNS SPF lookup MTA options.

39.18.1 MAIL FROM domain blocking by IP address (`blocked_mail_from_ips`)

The introduction of DNS wildcard entries in the COM and ORG top level domains which occurred in September 2003 had severely limited the effectiveness of the `mailfromdnsverify` channel option. As of the 6.1-0.01 release of the MTA, the `mailfromdnsverify` channel option code was modified to address this. When the DNS returns one or more A records (which would normally be considered a "success" and the message would be allowed in), their values are compared against the domain literals specified by the `blocked_mail_from_ips` MTA option. If a match is found, then the domain is considered to be invalid. Thus in order to restore useful behavior to the `mailfromdnsverify` channel option, the current correct setting of this option is:

```
blocked_mail_from_ips=[64.94.110.11]
```

39.18.2 Notification message MTA options: `return_envelope` (bitmask)

The `return_envelope` MTA option takes a bitmask value.

Bit 0 (value = 1) controls whether or not [return notifications generated by the MTA](#) are written with a blank envelope address *vs.* with the [address of the local postmaster](#). Setting the bit forces the use of the local postmaster address, while clearing the bit forces the use of a blank address. Note that the use of a blank address is mandated by [RFC 1123](#). However, some systems do not handle blank envelope From addresses properly 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. Again, this is used to accommodate non-compliant systems that don't conform to [RFC 821](#), [RFC 822](#), or [RFC 1123](#).

Bit 2 (value = 4) controls whether or not the MTA checks that any (non-empty) envelope From address matches (rewrites to) an MTA channel.

Setting bit 3 (value = 8) is the global (for all channels) equivalent of setting the [mailfromdnsverify channel option](#): it controls whether or not the MTA checks that the domain in the envelope From address resolves in the DNS. That is, setting the bit causes the MTA to require that a DNS entry can be found corresponding to the domain in the envelope From address; but the type of DNS entry does not matter.

Setting bit 4 (value = 16) causes the MTA to enforce that if the envelope From address claims a local domain name, the envelope From address must correspond to a user address (user alias).

New in 8.0, bit 6 (value = 64) modifies the effect of setting bit 3 (value = 8) on domain validity checks. With both these bits set, if the domain in the MAIL FROM address corresponds to a null MX domain, that address will be rejected as invalid. That is, setting bit 6 causes the bit 3 domain check to also implement support for draft-delany-nullmx-01.txt.

Note also that the [returnenvelope channel option](#) can be used to impose these sorts of control on a per-channel basis.

39.19 Error text and error interpretation MTA options

The MTA has a number of options affecting error interpretation, and allowing setting (customizing) of error text.

Regarding errors connecting to a spam/virus filter package, see also the [spamfilterN_optional](#) MTA options

39.19.1 Error text and error interpretation MTA options: `access_errors` (0 or 1)

As of JES MS 6.2, if `access_errors` is set to 0 (the default), then when a recipient address encounters a [recipient address * _ACCESS mapping table](#) access failure (that does not supply explicit rejection text of its own), the MTA will report it as if the error were an "unknown host" error. That is, the text of the [error_text_unknown_host](#) MTA option will be used, so by default the error will be reported as an "unknown host or domain" error, corresponding to the SMTP error:

550 5.7.1 unknown host or domain: *recipient-address*

This is the same error that would be reported if the address were simply illegal. Although confusing, this usage nevertheless provides an important element of security in circumstances where information about access restrictions should not be revealed. Setting `access_errors` to 1 will override this default and provide a more descriptive default error text, as specified by the [error_text_access_failure](#) MTA option, defaulting to 5.7.1 you are not allowed to use this address, corresponding to the SMTP error:

550 5.7.1 you are not allowed to use this address: *recipient-address*

But in any case, the setting of `access_errors` merely controls the default error text issued for [recipient address * _ACCESS mapping table](#) rejections; entries that perform rejections may override such default rejection text by supplying their own explicit rejection text. Prior to JES MS 6.2, this `access_errors` option did not affect the default text used for recipient address `* _ACCESS mapping table $N` rejections, which was instead the [error_text_permanent_failure](#) text, normally "unknown host or domain".

This option also controls the default error text issued when a spam/virus filter package rejects a recipient address with an other-than-temporary rejection.

This option also, in versions prior to 7.0-0.04, affected the now obsolete-and-removed feature whereby MTA provided facilities to restrict access to channels on the basis of group ids on UNIX (the analogue of rightslist identifiers in PMDF for OpenVMS). That is, in versions prior to 7.0-0.04, the error text issued in cases of address rejections due to access failures due to non-matching group id would also be affected by this option. Indeed, prior to JES MS 6.2, this (control of the error text due to group id mismatch) was the only purpose and only effect of this option.

39.19.2 error_text MTA options

The `error_text_*` options specify error text describing various error conditions; see the [Table of error_text_* MTA options](#) for details. Not all of the error responses potentially emitted by the MTA are configurable. In general, only error conditions that can be considered more or less local, or more or less proprietary to the MTA, such as invalid address conditions, user status problems such as a user being disabled or over-quota, attempts to exceed local message size limits, Sieve filter syntax errors, *etc.*, are configurable. As a rule of thumb, error conditions that arise solely in the case of a message addressed to a "local" user have configurable error text, on the presumption that customized explanations may be useful and are likely to be comprehensible to someone who corresponds with a "local" user.

However, error conditions that are more fundamental to the SMTP protocol, that more naturally arise when the MTA is performing a function of pure SMTP relaying, generally do not have configurable error text; in such cases where an error response may well be going back to some remote user who has no connection (not even as a correspondent) with "local" users, the MTA always emits its own standard, technically precise, error response.

Keep in mind that all the `error_text_*` error text may potentially be emitted as SMTP error response text. Thus the values of all these options must conform to the requirements of SMTP error response text. In particular, they are constrained to be in the US-ASCII character set: the MTA will convert any eight bit characters in such option values into the dollar character, `$`. Also, SMTP responses are limited by the SMTP line length limit (998 characters, not including the final CR/LF, and not including the leading numeric error code and extended error code).

Table 39.16 error_text_* MTA options

Option	SMTP code	Extended code	Default string used when option is not set	Meaning and notes
error_text_unknown_host	550 or 450*	5.1.2	unknown host or domain	Specifies the error text issued when, for instance, the domain in an address does not rewrite to any channel. (In particular, prior to Messaging Server 7.0.5, assuming that no "catch-all" or "." rewrite rule had been specified, an attempt to submit a message addressed to a top-level Internet domain not specified in the <code>internet.rules</code> file would result in this error. As of Messaging Server 7.0.5, the <code>internet.rules</code> file was modified to consist of solely a "." rule which consults the set of top level domain names from the <code>tlDs.txt</code> file -- so in Messaging Server 7.0.5 or later, an out-of-date <code>tlDs.txt</code> file or an attempt to submit a message to a top-level Internet domain not specified in the <code>tlDs.txt</code> file will result in this error.)
error_text_unknown_user	550 or 450*	5.1.1	unknown or illegal user	This error will be returned in cases such as illegal characters (or no characters) in a <code>uid</code> found during an <code>alias_urlN</code> lookup, or for users set to "native" (UNIX mailbox) delivery who do not have a UNIX account.
error_text_unknown_alias	550 or 450*	5.1.1	unknown or illegal alias	This error will be returned if an address that matches (rewrites to) a channel marked with the <code>viaaliasrequired</code> channel option is not found as an alias. In particular, this is normally the error that will be returned in the case of non-existent "user" addresses in a domain hosted on a Messaging Server system, unless a domain is configured with special handling for "non-existent user" addresses such as a "catch-all" address or a <code>mailRoutingSmartHost</code> .
error_text_access_failure	550	5.7.1	you are not allowed to use this address	This error text is not normally used unless the <code>access_errors</code> MTA option has been set. In that case, this error will be returned when an address cannot be submitted due to a group identifier on the destination channel, or due to <code>*_ACCESS mapping table</code> rejections.
error_text_alias_locked	452	4.2.0	list is currently reserved and locked	This error is returned if an attempt to look up an alias in the alias file finds the alias file locked, or if an attempt to access a list file (such as an <code>[AUTH_LIST]</code> file) finds the list file locked.
error_text_alias_auth	530 or 550++	5.7.1	you are not allowed to use this list	This error is returned when, for instance, a list posting authorization check fails. (See also the discussion of access errors in the discussions of the <code>ldap_reject_text</code> MTA option and <code>alias_error_text</code> alias option.)
error_text_alias_fileerror	452 or 550+	4.5.0	error opening file/ URL referenced by alias	This error is returned when a file or URL referenced by an alias cannot be opened; (that is, it exists but cannot be opened---compare with <code>error_text_alias_fileexist</code>).
error_text_alias_fileexist	452 or 550+	4.5.0	nonexistent file referenced by alias	This error is returned when a file referenced by an alias does not exist.
error_text_alias_temp	452	4.0.0	temporary error returned by alias expansion	This error is returned in cases, for instance, of temporary LDAP errors attempting to lookup an alias, such as the LDAP server not responding. It is also returned if a user entry that requires a <code>mailHost</code> attribute (not all user entries do) is lacking the attribute.
error_text_send_remote_error	550	5.6.1	no protocol to SEND/SAML	This error is returned in cases of an attempt to submit a message for direct broadcast (SEND) or for

				direct broadcast and e-mail (SAML) to a "local" user address that includes explicit routing characters (@, %, or !).
error_text_send_unknown_error	550	5.5.5	do not know how to SEND/SAML	This error will be returned if attempting to send to a destination (channel) that does not support SEND/SAML functionality.
error_text_block_over	550	5.2.3	channel limit of %d kilobytes on message size exceeded	This error will be returned (in the case of SMTP attempted submissions, at the RCPT TO stage of the SMTP dialogue) if a message exceeds the intended destination channel's <code>blocklimit</code> channel keyword setting <i>and</i> the sending e-mail client used the SIZE SMTP extension on the MAIL FROM command to inform the MTA "up front" of the message size. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "channel limit on message size exceeded".) This default error text, or any configured error text, will be suffixed with a colon and the recipient address being rejected. Note that this error text is not used for cases of exceeding a channel <code>sourceblocklimit</code> option setting, or the <code>block_limit</code> MTA option setting. First off, in those cases the MTA can potentially advertise its size limit to a sending client before a message is ever even submitted, so potentially the MTA never actually rejects the message itself; instead potentially a client that supports the SMTP SIZE extension refrains from even trying to send the message (and generates some message back to the original sender itself). If a message is submitted to the MTA despite the MTA's possibly advertised size limit, then in cases of exceeding a <code>sourceblocklimit</code> or <code>block_limit</code> when the client has used the SIZE extension, then the error text "Message exceeds local size limit." is used, or if the client does not use the SIZE extension (or puts a falsely small value in SIZE) so that the MTA has to reject the message after all the DATA has been sent and the MTA has computed the message size itself, the <code>error_text_message_too_large</code> message is used instead.
error_text_line_over	550	5.2.3	channel limit of %d lines on message length exceeded	This error will be returned if a message exceeds the intended destination channel's <code>linelimit</code> channel option setting, and that fact is apparent when the recipient address(es) are being processed. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "channel limit on message length exceeded".) However, message line length is <i>not</i> normally apparent at recipient address processing time, but rather only apparent after the message body is processed; therefore this error text is not normally used. Note also that this error text is not used for cases of exceeding the <code>line_limit</code> MTA option setting. Thus normally the following non-configurable text is used: "a message <i>x</i> lines long exceeds the line limit of <i>y</i> lines computed for this transaction".
error_text_list_block_over	550	5.2.3	list limit of %d kilobytes on message size exceeded	This error is returned if a message exceeds a list's configured size limit (in MTA blocks---see the <code>block_size</code> MTA option), configured via the <code>[BLOCKLIMIT]</code> named mailing list parameter for a list defined in the <code>alias file</code> or <code>alias database</code> , or via the <code>alias_blocklimit</code> alias option (Unified Configuration), or configured via the user/group-level <code>mgrpMsgMaxSize</code> attribute (more precisely, the attribute named by the <code>ldap_maximum_message_size</code> MTA option) or domain-level <code>mailDomainMsgMaxBlocks</code>

error_text MTA options

				attribute (more precisely, the attribute named by the ldap_domain_attr_blocklimit MTA option) for groups and lists defined in LDAP. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "list limit on message size exceeded".)
error_text_list_line_over	550	5.2.3	list limit of %d lines on message length exceeded	This error is returned if a message exceeds a list's configured line limit, configured via the [LINELIMIT] named mailing list parameter for a list defined in the alias file or alias database , or via the alias_linelimit alias option (Unified Configuration), and that fact is apparent when recipient address(es) are being processed. But since message line length is not normally known that early during message processing. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "list limit on message length exceeded".)
error_text_user_block_over	550	5.2.3	user limit of %d kilobytes on message size exceeded	This error is returned if a message exceeds the maximum message size (in MTA blocks--see the block_size MTA option) that a user may receive, as configured via the [BLOCKLIMIT] named parameter for aliases in the alias file or alias database , or via the alias_blocklimit alias option (Unified Configuration), or via the user-level <code>mailMsgMaxBlocks</code> attribute (more precisely, the attribute named by the ldap_blocklimit MTA option) or domain-level <code>mailDomainMsgMaxBlocks</code> attribute (more precisely, the attribute named by the ldap_domain_attr_blocklimit MTA option) for users defined in LDAP. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "user limit on message size exceeded".)
error_text_user_line_over	550	5.2.3	user limit of %d lines on message length exceeded	This error is returned if a message exceeds a user's configured line limit, configured via the [LINELIMIT] named parameter for an alias defined in the alias file or alias database , or via the alias_linelimit alias option in Unified Configuration, and that fact is apparent when recipient address(es) are being processed. But since message line length is not normally known that early during message processing, instead normally the non-configurable general error text "a message x lines long exceeds the line limit of y lines computed for this transaction" is used. As of 7.0.5, a %d, if present, will be replaced with the actual limit value. (Prior to 7.0.5, the default text was "user limit on message length exceeded".)
error_text_message_too_large	550	5.3.4	a message size of %d kilobytes exceeds the size limit of %d kilobytes computed for this transaction	New in 7.0.5 - this error message was hard coded in previous releases. This error will be returned (in the case of SMTP attempted submissions, at the end of the data transfer stage of the SMTP dialogue) if a message exceeds computed size limit for the transaction. The first %d, if present, will be replaced with the estimated message size and the second with the computed limit, both in units of MTA blocks .
error_text_message_too_long	550	5.3.4	a message %d lines long exceeds the line limit of %d lines computed for this transaction	New in 7.0.5 - this error message was hard coded in previous releases. This error will be returned (in the case of SMTP attempted submissions, at the end of the data transfer stage of the SMTP dialogue) if a message exceeds computed line limit for the transaction. The first %d, if present, will be replaced with the estimated number of message lines and the second with the computed limit.

error_text_insufficient_disk	452	4.3.4	message exceeds disk space available at this time	New in 7.0.5 - this error message was hard coded in previous releases. This error will be returned (in the case of SMTP attempted submissions, at the end of the data transfer stage of the SMTP dialogue) if the storage requirements for the message exceed the amount of disk space available.
error_text_wrong_account	550	5.7.17	account information on file is older than actual user account	This error is returned if an RRVS check on the account fails; see checkrrvs
error_text_wrong_domain	550	5.7.18	domain owner has changed	This error is returned if an RRVS check on the domain fails; see checkrrvs
error_text_recipient_over	452 or 550+	4.2.3	too many recipients specified	This error is returned if a message exceeds any configured limit on recipients, that is, exceeding a channel recipientlimit keyword setting, a FROM_ACCESS mapping table \$\$ recipient limit, a domain recipient limit (see the ldap_domain_attr_recipientlimit MTA option), or a user recipient limit (see the ldap_recipientlimit MTA option).
error_text_sieve_access	452	4.7.1	sieve filter access error	This error is returned when the MTA cannot open a recipient user Sieve filter file.
error_text_sieve_syntax	452	4.7.1	sieve filter syntax error	This error is returned when there is a syntax error in (or trouble in reading) a recipient user Sieve filter file.
error_text_disabled_user	550 or 450*	5.2.1	user disabled; cannot receive new mail	
error_text_disabled_alias	550 or 450*	5.2.1	alias disabled; cannot receive new mail	
error_text_over_quota	452 or 550+	4.2.2	user over quota; cannot receive new mail	This the error returned, for instance by the SMTP server, when the MTA detects that either a user's personal status (inetUserStatus or mailUserStatus) or their domain status (mailDomainStatus) is overquota during alias expansion. But note that once a message is in a final delivery channel (ims-ms or tcp_lmtp*) well past alias expansion processing, then the handling of messages to overquota users, and the error message returned, is controlled by the Message Store's configuration of overquota message handling , including the Message Store's grace period configuration , and the Message Store uses the IMAP over quota text as its error text in reports on overquota messages.
error_text_temporary_failure	452		unknown host or domain	Note that this error text for certain generic temporary failures defaults to the same error text used also in cases of clear-cut "bad" domain names, error_text_unknown_host , as well as in cases of generic permanent failures, error_text_permanent_failure .
error_text_permanent_failure	550 or 530++		unknown host or domain	Note that this error text for generic permanent failures defaults to the same error text used also in cases of clear-cut "bad" domain names, error_text_unknown_host , as well as in cases of generic temporary failures, error_text_temporary_failure .
error_text_illegal_8bit	553	5.1.3	illegal 8bit characters in address	Incorrect eight bit data present in an address.
error_text_disallowed_8bit	553	5.1.0	8bit characters in address not	Eight bit data present in an address when eight bit is not allowed.

error_text MTA options

			allowed in this context	
error_text_receipt_it	250	2.0.0	message accepted for list expansion processing	This option specifies the text used (by default "message accepted for list expansion") by the MTA when generating a delivery receipt (a notification message) to let a sender know that their message has gotten to the point of being expanded to a list. Note that the NOTARY specification (RFC 3461) explicitly requires that delivery receipt requests to mailing lists be responded to at the list expansion step; see Section 5.2.7.1 of RFC 3461 (which updates Section 6.2.7.1 of RFC 1891).
error_text_inactive_user	452 or 550+	4.2.1	mailbox temporarily disabled	
error_text_inactive_group	452 or 550+	4.2.1	group temporarily disabled	
error_text_disabled_group	550	5.2.1	group disabled; cannot receive new mail	
error_text_deleted_user	550	5.1.6	recipient no longer on server	This error will be returned when a user has an inetUserStatus or mailUserStatus attribute, (more precisely, an attribute named by the ldap_user_status or ldap_user_mail_status MTA options) with value of "deleted" or "removed"
error_text_deleted_group	550	5.1.6	group no longer on server	This error will be returned when a group has an inetMailGroupStatus attribute with value of "deleted" or "removed"; or more precisely, if either of the attributes named by the ldap_group_status or ldap_group_mail_status MTA options has such a value
error_text_duplicate_addrs	553	5.1.4	duplicate/ambiguous directory match	The recipient address has matched multiple entries in the directory; this typically indicates that a directory configuration error has occurred.
error_text_spamfilter_error	451	4.7.1	filtering/scanning error	As of JES MS 6.3-0.01, a synonym for the new-in-6.3 error_text_spamfilter1_error MTA option; as of JES MS 6.3, obsolete and used only if error_text_spamfilter1_error is not set.
error_text_spamfilterN_error	451	4.7.1	filtering/scanning error	New in JES 5. The default error text to use when there is a problem attempting to use the Nth spam/virus filter package, if no more specific error text regarding the exact spam/virus filter package problem is available; N can have values in the range 1--8
error_text_brightmail_error	451	4.7.1	filtering/scanning error	This obsolete option is used only if the error_text_spamfilter1_error MTA option is not set
error_text_still_held	452	4.2.1	cannot reenqueue while still held	Default error text to use when there is an attempt to reenqueue to a recipient whose status is "hold"; for instance, an attempt to release a message from the hold channel when a recipient still has a personal or domain status of "hold"
error_text_empty_alias	550	5.2.4	alias failed to expand to any valid addresses	New in JES MS 6.1.
error_text_nosolicit	550 or 530++	5.7.1	solicitations of this type are not allowed	New in JES MS 6.2. Default solicitation violation rejection text, if no more specific rejection text is available.
error_text_srs_syntax	553	5.1.3	Syntax error in SRS/MUL address	

error_text_srs_timeout	550	5.7.1	SRS/MUL address has timed out	
error_text_srs_badhash	550	5.7.1	SRS/MUL address has a bad hash value	
error_text_spf_temperror_4	451	4.7.24	temporary error in SPF verification of MAIL FROM domain	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_temperror is set to 4, then this is error text to use when a temporary DNS error occurs attempting an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage; the domain name (inside parentheses) will be suffixed to the specified error text
error_text_spf_temperror_5	550	4.7.24	temporary error in SPF verification of MAIL FROM domain	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_temperror is set to 5, then this is error text to use when a temporary DNS error occurs attempting an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage; the domain name (inside parentheses) will be suffixed to the specified error text.
error_text_spf_permerror_4	451	5.7.24	permanent error in SPF verification of MAIL FROM domain	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_permerror is set to 4, then this is error text to use when a permanent DNS error occurs attempting an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage; the domain name (inside parentheses) will be suffixed to the specified error text.
error_text_spf_permerror_5	550	5.7.24	permanent error in SPF verification of MAIL FROM domain	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_permerror is set to 5, then this is error text to use when a permanent DNS error occurs attempting an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage; the domain name (inside parentheses) will be suffixed to the specified error text.
error_text_spf_fail_4	451	5.7.23	SPF verification of MAIL FROM domain failed	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_fail is set to 4, then this is the error text to use when an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage determines that the domain has failed to verify; the domain name (inside parentheses) will be suffixed to the specified error text. If additional explanation text is available, then it will also be suffixed (with a colon) after the domain name. So, for instance, the entire error could appear as: 451 5.7.23 SPF verification of MAIL FROM domain failed (domain): explanation or: 451 5.7.23 error_text_spf_fail_4 (domain): explanation
error_text_spf_fail_5	550	5.7.23	SPF verification of MAIL FROM domain failed	(New in JES 6.3 , but not taking effect until Messaging Server 8.0) If the MTA option spf_smtp_status_fail=5 is set, then this is the error text to use when an SPF lookup on the domain from the MAIL FROM at either the MAIL FROM or RCPT TO stage determines that the domain has failed to verify; the domain name

error_text MTA options

				<p>(inside parentheses) will be suffixed to the specified error text. If additional explanation text is available, then it will also be suffixed (with a colon) to the error text.</p> <p>So, for instance, the entire error could appear as: 550 5.7.23 SPF verification of MAIL FROM domain failed (domain): explanation or: 550 5.7.23 error_text_spf_fail_5 (domain): explanation</p>
error_text_spf_softfail_4	451	4.7.23	SPF verification of MAIL FROM domain soft failed (<i>domain</i>)	<p>(New in JES 6.3, but not taking effect until Messaging Server 8.0) This is the error text to use when an SPF lookup of the MAIL FROM domain, performed at MAIL FROM time (configured via use of spfmailfrom) or RCPT TO time (configured via use of spfrcptto), determines that the domain has a "soft" verification failure and the MTA is configured to treat such verification failures as temporary errors: either an SPF SoftFail was returned for the specific domain name and spf_smtp_status_softfail=4 is set, or an SPF SoftFail "all" was returned for domain names including the specific domain name and spf_smtp_status_softfail_all=4 is set. The domain name (inside parentheses) will be suffixed to the specified error text. Note that for SPF lookups performed at EHLO/HELO (spfhello) time, the error_text_spf_ehlo_softfail_4 text is used instead.</p>
error_text_spf_softfail_5	550	4.7.23	SPF verification of MAIL FROM domain soft failed	<p>(New in JES 6.3, but not taking effect until Messaging Server 8.0) This is the error text to use when an SPF lookup of the MAIL FROM domain, performed at MAIL FROM time (configured via use of spfmailfrom) or at RCPT TO time (configured via use of spfrcptto), determines that the domain has a "soft" verification failure and the MTA is configured to treat such verification failures as permanent errors: either an SPF SoftFail was returned for the specific domain name and spf_smtp_status_softfail=5 is set, or an SPF SoftFail "all" was returned for domain names including the specific domain name and spf_smtp_status_softfail_all=5 is set. The domain name (inside parentheses) will be suffixed to the specified error text. Note that for SPF lookups performed at EHLO/HELO (spfhello) time, the error_text_spf_ehlo_softfail_5 text is used instead.</p>
error_text_spf_ehlo_temperror_4	451	4.7.24	temporary error in SPF verification of EHLO/HELO domain	<p>(New in 8.0) If the MTA option spf_smtp_status_temperror is set to 4, then this is error text to use when a temporary DNS error occurs attempting an SPF lookup on the domain from the EHLO/HELO command.</p>
error_text_spf_helo_temperror_5	550	4.7.24	temporary error in SPF verification of EHLO/HELO domain	<p>(New in 8,0) If the MTA option spf_smtp_status_temperror is set to 5, then this is error text to use when a temporary DNS error occurs attempting an SPF lookup on the domain from the EHLO/HELO command.</p>
error_text_spf_helo_permerror_4	451	5.7.24	permanent error in SPF verification of EHLO/HELO domain	<p>(New in 8,0) If the MTA option spf_smtp_status_permerror is set to 4, then this is error text to use when a permanent DNS error occurs attempting an SPF lookup on the domain from the EHLO/HELO command</p>
error_text_spf_ehlo_permerror_5	550	5.7.24	permanent error in SPF verification of EHLO/HELO domain	<p>(New in 8.0) If the MTA option spf_smtp_status_permerror is set to 5, then this is error text to use when a permanent DNS</p>

				error occurs attempting an SPF lookup on the domain from the EHLO/HELO command .
error_text_spf_ehlo_fail_4	451	5.7.23	SPF verification of EHLO/HELO domain failed	<p>(New in 8.0) If the MTA option spf_smtp_status_fail is set to 4, then this is the error text to use when an SPF lookup of the domain from the EHLO/HELO command determines that the domain has failed to verify. If additional explanation text is available, then it will be suffixed (with a colon) to the error text.</p> <p>So, for instance, the entire error could appear (at the EHLO/HELO command stage) as: 451 5.7.23 SPF verification of EHLO/HELO domain failed: explanation or: 451 5.7.23 error_text_spf_ehlo_fail_4: explanation</p>
error_text_spf_ehlo_fail_5	550	5.7.23	SPF verification of EHLO/HELO domain failed	<p>(New in 8.0) If the MTA option spf_smtp_status_fail=5 is set, then this is the error text to use when an SPF lookup of the EHLO/HELO domain determines that the domain has failed to verify. If additional explanation text is available, then it will be suffixed (with a colon) to the error text.</p> <p>So, for instance, the entire error could appear as: 550 5.7.23 SPF verification of EHLO/HELO domain failed: explanation or: 550 5.7.23 error_text_spf_ehlo_fail_5: explanation</p>
error_text_spf_ehlo_softfail_4	451	4.7.23	SPF verification of EHLO/HELO domain soft failed	<p>(New in 8.0) This is the error text to use when an SPF lookup of the EHLO/HELO domain name (configured via use of spfhello), determines that the domain has a "soft" verification failure and the MTA is configured to treat such verification failures as temporary errors: either an SPF SoftFail was returned for the specific domain name and spf_smtp_status_softfail=4 is set, or an SPF SoftFail "all" was returned for domain names including the specific domain name and spf_smtp_status_softfail_all=4 is set. If additional explanation text is available, then it will be suffixed (with a colon) to the error text.</p>
error_text_spf_ehlo_softfail_5	550	4.7.23	SPF verification of MAIL FROM domain soft failed	<p>(New in 8.0) This is the error text to use when an SPF lookup of the MAIL FROM domain, performed at RCPT TO time (configured via use of spfrcptto), determines that the domain has a "soft" verification failure and the MTA is configured to treat such verification failures as permanent errors: either an SPF SoftFail was returned for the specific domain name and spf_smtp_status_softfail=5 is set, or an SPF SoftFail "all" was returned for domain names including the specific domain name and spf_smtp_status_softfail_all=5 is set. If additional explanation text is available, then it will be suffixed (with a colon) to the error text.</p>
error_text_mailfromdnsverify	550 or 450	5.1.8 or 4.1.8	invalid/host-not-in-DNS return address not allowed	<p>(New in JES MS 6.3) The 450 4.1.8 error is returned for all cases of DNS verification lookup "failures" other than HOST_NOT_FOUND; hence DNS lookup difficulties such as DNS server failure to respond will result in this error. Normally, the 550 5.1.8 error is returned when a DNS verification lookup returns a definitive HOST_NOT_FOUND error. However, if bit 5/value 32 of the returnenvelope channel option is set, then HOST_NOT_FOUND will also result in a temporary 450 4.1.8 error rather than the permanent 550 5.1.8 rejection.</p>

error_text MTA options

error_text_null_mx	521 or 550**	5.1.10 or 5.7.26**	host/domain does not accept mail	(New in 8.0) This error is returned when the domain associated with a recipient address resolves to a so-called "null MX". This error message is also returned when bit 3 (value 8) and bit 6 (value 64) of the returnenvelope channel option or the return_envelope MTA option are set and a domain with a "null MX" appears in the envelope sender (MAIL FROM) address.
error_text_invalid_return_address	550	5.1.7	invalid/unroutable return address not allowed	(New in JES MS 6.3) The 550 5.1.7 error is returned if bit 2 (value 4) of the returnenvelope channel option is set and rewriting of the MAIL FROM failed to match any channel.
error_text_unknown_return_address	550	5.1.8	invalid/no-such-user return address	(New in JES MS 6.3) The 550 5.1.8 error is returned if bit 4 (value 16) of the returnenvelope channel option is set and the MAIL FROM address is local but could not be resolved to any known user.
error_text_accepted_return_address	250	2.5.0	return address invalid/unroutable but accepted anyway	(New in JES 6.3) This is not, properly speaking, an error - this message is returned when an invalid MAIL FROM address is given but accepted by the MTA anyway.
error_text_source_sieve_access	450	4.3.0	source channel sieve filter access error	(New in JES 6.3) This error is returned when the MTA cannot open a source channel Sieve filter file .
error_text_source_sieve_syntax	450	4.3.0	source channel sieve filter syntax error:	(New in JES 6.3) This error is returned when there is a syntax error in (or trouble in reading) a source channel Sieve filter .
error_text_source_sieve_authorization	450	4.3.0	source channel sieve filter authorization error	(New in JES 6.3) Currently unused.
error_text_transaction_limit_exceeded	450	4.5.3	number of transactions exceeds allowed maximum	(New in JES 6.3) Error returned at MAIL FROM when the channel transactionlimit option, specifying the maximum number of transactions allowed in the session, is exceeded.
error_text_insufficient_queue_space	450	4.3.1	insufficient free queue space available	(New in JES 6.3) Issued in response to a MAIL FROM: command if the free disk space available to the MTA in the MTA's queue area dips below 10 MTA blocks
error_text_temporary_write_error	451	4.4.5	error writing message temporary file	(New in JES 6.3) The SMTP server prefixes this error text with the message: 451 4.4.5 Error writing message temporaries - An internal channel such as the reprocess channel would record this error text in its delivery history , and in its "Q" record.
error_text_smtp_lines_too_long	554	5.6.0	lines longer than SMTP allows encountered; message rejected	(New in JES MS 6.3) Issued when rejectsmtplonglines is in effect, and a line longer than 998 characters (not including the SMTP CRLF line terminator) is seen in the message data.
error_text_unnegotiated_eightbit	554	5.6.0	message contains unnegotiated 8bit	(New in JES MS 6.3) Issued when a source TCP/IP channel has eightstrict or utf8strict set, and the incoming message contains unnegotiated eight bit data.
error_text_mls_access_failure	550	5.7.1	security access check failure	(New in 7.0) This restricted option is currently unused.
error_text_spare_error				Obsolete (does not exist) as of JES MS 6.2.
error_text_spare1_error				(New in JES MS 6.2) This option provides a spare slot so a new settable error message can be added to an existing release. Use of this option is restricted.
error_text_spare2_error				(New in JES MS 6.2) This option provides a spare slot so a new settable error message can be added to an existing release. Use of this option is restricted.

+ Whether the error code used is a temporary 4yz (the default) or a permanent 5yz error code is controlled by the [use_permanent_error](#) MTA option.

++ In place of the usual 550 error code, the 530 error code is used when the problem relates to security: as for instance failure to properly authenticate (successfully use SMTP AUTH) when authentication is required.

+++ Errors at MAIL FROM: stage use 450; errors at RCPT TO: stage use 452.

* (Added in 8.0.) Whether the error code used is a permanent 5yz (the default) or a temporary 4yz error code is controlled by the [use_temporary_error](#) MTA option.

** (Added in 8.0) Error regarding null MX for a recipient uses 521 5.1.10; error regarding null MX for a sender uses 550 5.7.26.

Also note that errors authenticating (errors attempting SMTP AUTH use) are a separate category of error type, returning hard-coded error text. (So for instance the [error_text_disabled_user](#) option discussed above is relevant to attempts by the MTA to verify that the user is a currently valid recipient; for instance, that error could be returned as an SMTP rejection of that user's address as an envelope recipient address. But an attempt by that same disabled user to submit a message using SMTP AUTH to authenticate would fail authentication and result in a different error, discussed in the table [MTA AUTH errors](#).) Note that for security reasons, a number of different underlying error conditions cause the same error text to be returned in the SMTP rejection, while more specific details are available in the reason field of MTA connection transaction logging if the MTA option [log_reason](#) is enabled.

Table 39.17 MTA AUTH errors

SASL error or code	SMTP code	Extended code	Basic+ SMTP error text	log_reason text
LDAP server unavailable/unresponsive for authentication	450	4.3.0	SASL initialization failed; server unavailable.	
LDAP server unavailable/unresponsive for authentication	454	4.7.0	Authentication server unavailable <i>sasl-err-detail</i> .	
SMTP AUTH already successfully performed.	503	5.7.0	AUTH command already issued.	
No <code>maysaslserver</code> or <code>mustsaslserver</code> enabled on channel.	533	5.7.1	AUTH command is not enabled.	
Argument right of = fails to BASE64 decode.	501	5.7.0	Cannot decode BASE64.	
SMTP AUTH not permitted now that message submission has begun.	503	5.7.1	Mail transaction already in progress.	
Argument on new line fails to BASE64 decode.	501	5.7.0	Cannot decode BASE64	
?	501	5.7.0	AUTH operation aborted by client.	Client aborted AUTH operation

use_permanent_error MTA option

SASL_OK	235	2.7.0	<i>mechanism</i> authentication successful.	authentication successful++
SASL_NOMECH	504	5.5.4	Unrecognized authentication type	Unrecognized authentication type
SASL_BADPROT	501	5.5.0	Invalid input	Invalid input
SASL_NOUSER	535	5.7.8	Bad username or password	No such user
SASL_TOOWEAK	535	5.7.8	Bad username or password	Authentication mechanism is too weak
SASL_BADAUTH	535	5.7.8	Bad username or password	Bad password
SASL_NOAUTHZ	535	5.7.8	Authorization failure	Authorization failure
SASL_ENCRYPT	523	5.7.10	Encryption needed to use mechanism	Encryption needed for mechanism
SASL_EXPIRED	524	5.7.11	Password expired, has to be reset	Password expired; has to be reset
SASL_DISABLED with mailUserStatus: inactive	525	5.7.13	Account disabled	Account disabled (inactive)
SASL_DISABLED with mailUserStatus: hold	525	5.7.13	Account disabled	Account disabled (hold)
SASL_UNAVAIL	454	4.7.0	Authentication server unavailable	Authentication server unavailable
SASL_TRYAGAIN	454	4.7.0	Try again later <i>sasl-err-detail</i>	Try again later
SASL_TRANS	422	4.7.12	Try changing your password	Transition password needed
Default for other errors	500	5.7.0	Unknown AUTH error <i>sasl-err-detail</i>	

+ Additional detail error text potentially may be suffixed within parentheses for error cases other than a client abort of the AUTH attempt, or successful authentication.

++ New in 7.3-11.01 version; previously, the [log_reason](#) field was the empty string for this success case

39.19.3 Error text and error interpretation MTA options: use_permanent_error (bitmask)

The use_permanent_error MTA option controls whether certain error conditions normally considered to be temporary are instead interpreted as permanent errors; that is, this option controls whether certain error conditions result in immediately bouncing (returning) messages instead of retaining them for further delivery retries. The option takes a bit encoded integer argument, with the individual bits each controlling an individual error condition. The default is 0.

Table 39.18 use_permanent_error MTA option bits

Bit	Value	Usage
0	1	If set, causes a permanent rather than a temporary error to be returned for the error_text_inactive_user case; that is, if set, then in the case of users with an "inactive" status, the (permanent) error "550 4.2.1 mailbox temporarily disabled" is issued rather than the (temporary) error "450 4.2.1 mailbox temporarily disabled". Note that the actual text issued in the error (by default "mailbox temporarily disabled") may be customized using the error_text_inactive_user MTA option.
1	2	If set, causes a permanent rather than a temporary error to be returned for the error_text_inactive_group case.
2	4	If set, causes a permanent rather than a temporary error to be returned for errors corresponding to the error_text_over_quota case.
3	8	If set, causes a permanent rather than a temporary error to be returned for the error_text_alias_fileerror and error_text_alias_fileexist cases.
4	16	If set, causes a permanent rather than a temporary error to be returned for the error_text_recipient_over case.

The [usepermanenterror](#) channel option may be used on a per-source-channel basis to override the MTA level `mta.use_permanent_error` setting.

39.19.4 Error text and error interpretation MTA options: use_temporary_error (bitmask)

(New in MS 8.0) The `use_temporary_error` MTA option controls whether certain error conditions normally considered to be permanent are instead interpreted as temporary errors; that is, this option controls whether certain error conditions result retaining messages for further delivery retries instead of immediately bouncing (returning) them. The option takes a bit encoded integer argument, with the individual bits each controlling an individual error condition. The default is 0.

Table 39.19 use_temporary_error MTA option bits

Bit	Value	Usage
0	1	If set, causes a temporary rather than a permanent error to be returned for the error_text_disabled_user case.
1	2	If set, causes a temporary rather than a permanent error to be returned for the error_text_disabled_alias case.
2	4	If set, causes a temporary rather than a permanent error to be returned for errors corresponding to the error_text_unknown_alias case.
3	8	If set, causes a temporary rather than a permanent error to be returned for the error_text_unknown_host case.

4	16	If set, causes a temporary rather than a permanent error to be returned for the error_text_unknown_user case.
---	----	---

The [usetemporaryerror](#) channel option may be used to override the MTA level `mta.use_temporary_error` on a per-source-channel basis.

39.20 External filtering context MTA options

The `scan_*` MTA options set context for filtering performed by external-to-the-MTA components, such as when [imexpire makes use of Sieve rules or spam/virus filter packages](#) to perform message expiration.

For MTA options relating to the MTA's own Sieve filter use or Spam/virus filter package integration, see respectively the [Sieve filter MTA options](#) and [Spamfilter MTA options](#).

39.20.1 External filtering context MTA options: `scan_channel` (MTA channel name)

New in 7.0.5. The MTA's spam/virus filter package integration facility can be used in non-MTA (in particular, in non-channel) contexts, such as by applications or utilities such as [imexpire](#), where certain fields inherently available for message processing in the MTA context are not necessarily naturally available. The `scan_channel` MTA option specifies the default source channel for such scanning operations. The default is the `l` (lowercase "L") channel.

39.20.2 External filtering context MTA options: `scan_originator` (address)

New in 7.0.5. The MTA's spam/virus filter package integration facility can be used in non-MTA (in particular, in non-channel) contexts, such as by applications or utilities such as [imexpire](#), where certain fields inherently available for message processing in the MTA context are not necessarily naturally available. The `scan_originator` MTA option specifies the MAIL FROM address that's used for such scanning operations. (There is no envelope in this situation, but the MTA requires one, so a minimal pseudo envelope has to be constructed.) The default is the empty string.

39.20.3 External filtering context MTA options: `scan_recipient` (address)

New in 7.0.5. The MTA's spam/virus filter package integration facility can be used in non-MTA (in particular, in non-channel) contexts, such as by applications or utilities such as [imexpire](#), where certain fields inherently available for message processing in the MTA context are not necessarily naturally available. The `scan_recipient` MTA option specifies the RCPT TO address that's used for such scanning operations. (There is no envelope in this situation and thus no recipient, but the MTA requires one, so a pseudo recipient must be claimed.) The default is "postmaster".

39.21 File format MTA options

There are a number of MTA options affecting the format of various MTA files, or the handling of message files or log files. In particular, some of these MTA options can have performance implications.

See also the [file_member_size](#) MTA option, which sets a limit for the number of configuration files the MTA may use.

See also the [os_debug](#) MTA option which enables some low-level debugging of MTA file handling.

39.21.1 MTA enqueue buffering ([buffer_size](#))

The [buffer_size](#) MTA option sets the buffer size (for PMDF in blocks, as defined via the [block_size](#) MTA option; for the Messaging Server MTA, [buffer_size](#) is literally the number of bytes) used when writing message files to the MTA's queues (as well as when writing temporary files when [max_internal_blocks](#) has been exceeded). The default value is 8192. The minimum allowed value is 512; the maximum allowed value is $512 \times 512 = 262144$.

Note that this MTA option is completely different from the [BUFFER_SIZE](#) TCP/IP-channel-specific option (which is instead more closely analogous to the [max_internal_blocks](#) MTA option).

39.21.2 File format MTA options: [cache_magic](#) (integer)

The **OBSOLETE** [cache_magic](#) MTA option was used in PMDF versions to control the order of sorting (for processing purposes) of old message files on disk. The default was 87654321.

This option is **OBSOLETE** and has no effect in modern MTA versions.

39.21.3 File format MTA options: [cbt](#) (0 or 1; OpenVMS only)

The **OBSOLETE** [cbt](#) MTA option was used in PMDF versions on OpenVMS to control the use of "contiguous best try" filesystem storage. Setting the option to 1 sets the corresponding bit in the file attributes block, or FAB.

This option is **OBSOLETE** and has no effect in modern MTA versions.

39.21.4 File format and file handling options ([comment_chars](#))

The [comment_chars](#) MTA option controls what characters are taken to signal a comment when they appear in the first column of various MTA input files. The value of this option takes the form of a list of ASCII character values in decimal. In Unified Configuration, such a list is space separated:

```
msconfig> show -default mta.comment_chars
mta.comment_chars: 33 59
```

In legacy configuration, the default was the list {33, 59}. With either representation, this specifies exclamation points and semicolons as comment introduction characters.

As of 7.0.5, note that the `comment_chars` MTA option only truly controls which character(s) *in addition to* semicolon (ASCII value 59) represent comment introduction characters, since regardless of whether or not it is explicitly set (or shows up in the value of `comment_chars`), the semicolon will always be treated as being set in `comment_chars`.

Note that the `comment_chars` option does not apply when reading the [legacy configuration MTA option file](#) itself as this file is read before `comment_chars` has even been seen. For this file, the following comment characters are "hard-coded": "!", ";", and "#".

39.21.5 Debug MTA options: `debug_flush`

As of Messaging Server 7.1, a.k.a. Messaging Server 7.0-3.01, the `debug_flush` MTA option causes certain debug output to get immediately flushed to disk. This is applicable for many MTA components, including typical [channel debug output](#), but it is especially relevant and noticeable for long-running components such as the [SMTP server](#), and [Job Controller](#). The flush-to-disk-log-file of the debug output may incur a bit of a performance penalty, but tends to be more convenient for debugging purposes. The default is that such debug flushing is not enabled (`debug_flush = 0`).

As of 7.0.5, the `debug_flush` MTA option can also cause flushing of [Dispatcherdebug](#) output.

39.21.6 File reading during dequeue (`dequeue_map`)

The `dequeue_map` option controls whether files are mapping into memory when dequeuing. The default is 1 (true), meaning that files are so mapped. Disabling this, by setting this option to 0 (false), can be expected to have a detrimental effect on performance (and will likely cause the [ims-ms channel](#) to abort and core).

Caution: Use of this option is **RESTRICTED**.

39.21.7 File format MTA options: `fdirectory` (0 or 1; OpenVMS only)

`fdirectory` is an OpenVMS only MTA option.

39.21.8 File format MTA options: `fsync` (0 or 1)

RESTRICTED.

The `fsync` MTA option may be used to cause the MTA to use the `fsync` function (UNIX) to flush disk output when closing a message file. If such flushing is not performed explicitly by the MTA, it is left up to the O/S to perform on its own timetable; potentially, if a system crashes at just the wrong moment, messages not yet synched to disk could be lost. The tradeoff, however, is that performing explicit flushing for every message incurs a performance penalty. `fsync=1`, meaning that flushes are performed explicitly by the MTA, is the default, ensuring message safety at the expense of a performance hit.

For the Message Store, somewhat analogous is the former store option [diskflushinterval](#) (default 15), and even more so the newer `*synclevel` store options such as

[messagesynclevel](#) (not advisable to set, other than in the special case of restoring from backup, per the iMS 5.3 Migration Guide).

39.21.9 File format MTA options: `log_alq` (integer)

OpenVMS only.

The `log_alq` MTA option specifies the default allocation quantity (in OpenVMS blocks) for the MTA message transaction log file, `mail.log_current`. The default value is 2000, or twice the `log_deq` value if `log_deq` has been explicitly set. On a busy system that is updating that log file frequently, increasing this value may provide increased efficiency.

39.21.10 File format MTA options: `log_deq` (integer)

OpenVMS only.

The `log_deq` option specifies the default extend quantity (in OpenVMS blocks) for the MTA message transaction log file, `mail.log_current`. The default value is 1000. On a busy system that is updating that log file frequently, increasing this value may provide increased efficiency.

39.21.11 File format MTA options: `max_internal_blocks` (integer)

The `max_internal_blocks` MTA option specifies how large (in MTA blocks---see the [block_size](#) MTA option) an incoming message the MTA will buffer entirely in memory; messages larger than this size will be written to temporary files (in the area specified by the [tmpdir](#) MTA option -- or the `IMTA_TMP` tailor file option, in legacy configuration). (Prior to JES JES MS 6.0, the area such files were written to was controlled by the now obsolete `IMTA_SCRATCH` tailor file option.) The default is 30.

For systems with lots of memory, increasing this value may provide a performance improvement.

Note that `max_internal_blocks` does not affect incoming LMTP messages; that is, it does not affect the [LMTP server's](#) handling of incoming messages. As of circa 6.2p5/6.3, the LMTP server will buffer in memory up to a default of 1,000,000 bytes (in JES MS 6.2p5 this number is hard-coded and not configurable) or the value set for the LMTP server [BUFFER_SIZE](#) TCP/IP-channel-specific option of an incoming message, and after that will buffer to a temporary file (`tmpdir` or `IMTA_TMP` file). Prior to this change, the LMTP server did not do in-memory buffering and instead always buffered to temporary files (in `IMTA_TMP`).

39.21.12 File format MTA options: `mm_mbc` (0-255)

OpenVMS only.

The `mm_mbc` MTA option sets the RMS RAB MBC field (the RMS disk block size) used when writing message files.

39.21.13 File format MTA options: `mm_mbf` (0-255)

OpenVMS only.

The `mm_mbf` MTA option sets the RMS RAB MBF field (the RMS multibuffer count) used when writing message files.

39.21.14 Notification message MTA options: `notary_quote` (1-127)

The `notary_quote` MTA option specifies the ASCII representation of the character that marks substitution sequences in `return_*.txt` files and `disposition_*.txt` files. It defaults to 25 (the ASCII position of the percent character) so substitutions are `%R`, `%u`, *etc.*, as listed in [return_*.txt file substitution sequences](#).

39.21.15 File format MTA options: `osync` (0 or 1)

RESTRICTED.

The `osync` MTA option controls whether MTA message queue file creation sets the `O_SYNC` flag. If set to 1, the `O_SYNC` flag is set when creating message queue file entries on disk. The default is 0. Setting `O_SYNC` may provide an increase in performance on ZFS file systems, but will degrade performance considerably on UFS.

39.21.16 `projectid` Option Under `mta`

The `project` MTA option overrides, for MTA use, the `projectid` base option. Thus the `projectid` MTA option specifies the numeric identifier the MTA uses when obtaining shared memory segments. This identifier is used in `ftok()` calls to generate a shared memory segment key. By default, the MTA uses the value of the `projectid` base option. If the `projectid` base option is not set, then the MTA defaults to using a value of 7; (in MS 7.4, if the `projectid` base option was not set the MTA defaulted to using a value of 1). Only the lowest eight bits of the value are significant.

39.21.17 File format MTA options: `queue_cache_mode` (integer)

RESTRICTED: The value 2 (the default) must be used for the Messaging Server MTA.

The `queue_cache_mode` MTA option tells the MTA the type of cache of message queue files to use. In particular, a value of 2, the default, means that the [Job Controller](#) is maintaining (in-memory) queue cache information. This option *must not* be set to any other value.

39.21.18 File format MTA options: `queue_cache_mode_3_files` (list of file paths)

RESTRICTED. Only relevant when the `queue_cache_mode` MTA option is set to a value of 3 (which is not currently supported).

39.21.19 File format MTA options: `use_text_databases` (bitmask)

The `use_text_databases` MTA option controls whether the [reverse "database"](#), the [general "database"](#), and the [forward "database"](#) are truly on-disk databases, or whether they are instead stored as in-memory structures. The default is 0, meaning that (unless the new in MS 8.0 [reverse_database_url](#), [general_database_url](#), or [forward_database_url](#) MTA options, respectively, are set) these three databases are accessed as true, on-disk databases. Setting a bit of `use_text_databases`, so that an in-memory structure is used, makes moot any setting of the corresponding, new in MS 8.0, `*_database_url` MTA option: the in-memory structure will be used in preference to consulting memcache.

If bit 0 (value 1) is set, then the `IMTA_TABLE:general.txt` file (`configroot/general.txt`) is read when the MTA configuration is initialized or reloaded, and the information from that file is stored in memory replacing all uses of the [general database](#).

If bit 1 (value 2) is set, then the `IMTA_TABLE:reverse.txt` file (`configroot/reverse.txt`) is read when the MTA configuration is initialized or reloaded, and the information from that file is stored in memory replacing all uses of the [reverse database](#).

If bit 2 (value 4) is set, then the `IMTA_TABLE:forward.txt` file (`configroot/forward.txt`) is read when the MTA configuration is initialized or reloaded, and the information from that file is stored in memory replacing all uses of the [forward database](#).

In particular, note that enabling use of such in-memory "databases" means that changes to the "database" (changes to the underlying text file source) require recompiling the configuration, or reloading the configuration, in order to get the change seen in a compiled configuration; see the `cnbuild` utility and the `reload` utility. As of JES MS 6.3-1.04, text databases support including other files via the `<` character, and comment lines (indicated by the presence of any of the [comment_chars](#) characters in column one).

39.22 Internal size MTA options

The MTA has a number of options that control internal MTA table sizes. These options do not normally need to be adjusted manually. Although they provide "starting points" (and maximums) for the sizes of various internal memory tables, the MTA will resize its internal tables as necessary when running. In particular, use of the `imsimta cnbuild` command will cause resizing to accommodate the current configuration size (and then all subsequently started processes will use the compiled configuration sizes). If a compiled configuration has not been created, then each process when it initially starts will need to do the resizing itself. Thus the penalty for not having "good" values set for these options is merely a little initial overhead—either overhead only for the `imsimta cnbuild` command when a compiled configuration is used, or overhead for each starting process when a compiled configuration is not used. In other words, the relevance of these options is limited mostly to "fine-tuning" memory usage, and providing a modest performance benefit for the `imsimta cnbuild` command (or for all starting processes if a compiled configuration is not in use).

For those who do wish to ensure that these options are set to values matching the MTA's real memory needs, using the `imsimta cnbuild` utility, that is, using a command such as the UNIX command

```
# imsimta cnbuild -noimage_file -maximum -option_file
```

is usually the best approach (letting the MTA tell you what values it needs, rather than trying to guess values and manually set them yourself). The above command will output MTA

options (stored in the MTA option file in an legacy configuration) with sizes adequate for the current configuration, plus some growth room. (Note that the above shown command does not actually compile the configuration using these sizes; that would require an additional `imsimta cnbuild` command. Rather, the above command determines appropriate sizing, and outputs corresponding MTA option values; such values could then be used in any subsequent MTA process invocation and in particular, in a subsequent `imsimta cnbuild` command.)

Since the MTA will resize its internal table sizes as needed, errors about exceeding table sizes are normally seen only if the MTA's more-or-less "hard" limits on resizing are reached. (The limits are established by the `maximum.dat` file and/or "hard" limits in the code.) And since the MTA's "hard" limits are *very* generous, exceeding the limits is usually an indication of either a configuration error of a type that has confused the MTA about the intended meaning of certain configuration inputs (for instance, an extraneous blank line in the rewrite rules, causing the MTA to attempt to interpret all remaining material as channel definitions), or configuration choices involving poor use of MTA facilities that would be better handled in an alternate manner (such as attempting to hard code many thousands of mapping table entries, rather than using a few general entries that do [general database](#) callouts for the specific fields). In particular, reaching the limits specified in the normal `maximum.dat` file is usually an indication of poor configuration choices; you should contact Oracle if you believe you wish to exceed those limits, as you may be better served by alternate configuration tactics.

Note: Historically, the default values, maximum size achievable through automatic resizing, and "hard" limits for these options have been prone to change (particularly increase) during and especially between releases, to a rather greater extent than for most other options. So although a diligent attempt has been made to provide correct numbers as of press time, the ongoing quest to improve performance and scalability may mean that these documented values become out-of-date.

39.22.1 Internal size MTA options: `alias_hash_size` (1-1000000)

The `alias_hash_size` MTA option sets the size of the alias hash table. This in turn is an upper limit on the number of aliases that can be defined in the alias file, or equivalently the number of [named alias groups in Unified Configuration](#). The default is 256; the maximum that the MTA will allow with normal, automatic resizing is 15,000 (controlled from the `maximum.dat` file); the "hard" maximum value allowed is 1,000,000. Attempts to set this option higher than 1,000,000 will result in an `mm_init` error, "ALIAS_HASH_SIZE exceeds maximum". Attempts to specify more aliases in the alias file than allowed by this option's maximum (normally the automatic, resize maximum, unless a higher maximum has been explicitly configured for this option) will result in an `mm_init` error, "no room in table for alias ...".

39.22.2 Internal size MTA options: `alias_member_size` (0-40000)

The `alias_member_size` MTA option controls the size of the index table that contains the list of alias translation value pointers. The total number of addresses on the right hand sides of all the alias definitions in the alias file, or addresses on the right hand sides of all [named alias groups in Unified Configuration](#), cannot exceed this value. The default is 320; the maximum allowed with normal, automatic resizing is 30,000 (controlled from the `maximum.dat` file); the

"hard" maximum allowed is 40,000. Attempts to set this option higher than 40,000 will result in an mm_init error, "ALIAS_MEMBER_SIZE exceeds maximum". Attempts to specify more alias translation values than allowed by the maximum for this option (normally, the automatic, resizing maximum, unless a higher maximum has been explicitly configured) will result in an mm_init error, "no room in alias member table for alias ...".

39.22.3 Internal size MTA options: channel_table_size (1-8192)

The channel_table_size MTA option controls the size of the channel table. The total number of channels in the configuration file, or the total number of [named channel groups in Unified Configuration](#), cannot exceed this value. The default is 256; the maximum that the MTA will allow with normal, automatic resizing is 2500 (controlled from the maximum.dat file); the "hard" maximum (only achievable via explicit manual configuration) is 8,192. Attempts to set this option higher than 8,192 will result in an mm_init error, "CHANNEL_TABLE_SIZE exceeds maximum". Attempts to define more channels in the MTA configuration than allowed by this option's maximum (normally the automatic, resize maximum, unless a higher maximum has been explicitly configured) will result in an mm_init error, "no room in channel table for ...".

39.22.4 Internal size MTA options: chunk_cache_limit (non-negative integer)

The chunk_cache_limit MTA option specifies the size of the cache of message body chunk storage buffers. This can affect the efficiency of processing large or multipart message bodies. The default is to allow up to 1024 buffers in the cache.

39.22.5 Internal size MTA options: circuitcheck_paths_size (0-256)

The circuitcheck_paths_size MTA option controls the size of the circuit check paths table, and thus the total number of circuit check configuration file entries. The default is 10; the maximum allowed with normal, automatic resizing is 128 (controlled from the maximum.dat file); the "hard" maximum is 256. Attempts to set this option higher than 256 will result in using the value 256 (with no error message).

39.22.6 Internal size MTA options: conversion_size (1-2000)

The conversion_size MTA option controls the size of the [conversion entry table](#), and thus the total number of conversion file entries (or entries in the [conversions](#) option in Unified Configuration) cannot exceed this number. The default is 32; the maximum allowed with normal, automatic resizing is 500 (controlled from the maximum.dat file); the "hard" maximum is 2,000. Attempts to set this option higher than 2,000 will result in using the value 2,000 (with no error message).

39.22.7 File format and file handling (describe_cache_limit)

The `describe_cache_limit` MTA option specifies the size of the cache of message part descriptions. This can affect the efficiency of processing large or multipart message bodies. The default is 256.

39.22.8 Internal size MTA options: `domain_hash_size` (1-1000000)

The `domain_hash_size` MTA option controls the size of the domain rewrite rules hash table. Each rewrite rule in the configuration file, or set in Unified Configuration, consumes one slot in this hash table, thus the number of rewrite rules cannot exceed this option's value. The default is 512; the maximum allowed with normal, automatic resizing is 16,384 (controlled from the `maximum.dat` file); the "hard" maximum number of rewrite rules allowed is 1,000,000. Attempts to set this option higher than 1,000,000 will result in an `mm_init` error, "DOMAIN_HASH_SIZE exceeds maximum". Attempts to specify more rewrite rules than allowed by this option's maximum (normally the automatic, resizing maximum, unless a higher maximum has been explicitly configured) will result in an `mm_init` error, "no room in rewrite rule table for ...".

39.22.9 Internal size MTA options: `file_member_size` (1-8192)

The `file_member_size` MTA option sets a limit for the number of configuration files the MTA may use. The default is 32; the "hard" maximum is 8,192. Attempts to set this option higher than 8,192 will result in an `mm_init` error, "FILE_MEMBER_SIZE exceeds maximum". Attempts to use more configuration files than allowed by this option's maximum (normally the automatic, resizing maximum, unless a higher maximum has been explicitly configured) will result in an `mm_init` error, "no room in file member table for file string ...".

39.22.10 Internal size MTA options: `forward_data_size` (1-1000000)

The `forward_data_size` MTA option sets the internal hash size for the forward (database) text file entries. The default is 64; the "hard" maximum is 1,000,000. Attempts to set this option higher than 1,000,000 will result in an `mm_init` error, "FORWARD_DATA_SIZE exceeds maximum". Attempts to specify more forward database entries than allowed by this option's maximum will result in an `mm_init` error, "no room in table for forward data ...".

39.22.11 Internal size MTA options: `fruits_size` (1-20000)

The `fruits_size` MTA option controls the total number of fruits allowed in the fruit validation table. The default is 110; the maximum value allowed is 20,000.

39.22.12 Internal size MTA options: `general_data_size` (1-2000000)

The `general_data_size` MTA option controls the internal hash size for the [general \(database\) text file entries](#). The default is 256; the "hard" as of 8.0, the maximum is 5,000,000;

the maximum changed from 1,000,000 in JES MS 6.2 to 2,000,000 for JES MS 6.3. Attempts to set this option higher than its hard maximum (1,000,000 or 2,000,000 or 5,000,000, depending upon version) will result in an mm_init error, "GENERAL_DATA_SIZE exceeds maximum". Attempts to specify more entries in the general database than this option's maximum will result in an mm_init error, "no room in table for general data ...".

39.22.13 Internal size MTA options: host_hash_size (1-1000000)

The host_hash_size MTA option controls the size of the channel hosts hash table. Each channel `host` specified on a channel definition in the MTA configuration file (both official hosts and aliases) consumes one slot in this hash table, so the total number of channel hosts cannot exceed the value specified. The default is 512; the maximum allowed with normal, automatic resizing is 16384 (controlled from the `maximum.dat` file); the "hard" maximum value allowed is 1,000,000. Attempts to set this option higher than 1,000,000 will result in an mm_init error, "HOST_HASH_SIZE exceeds maximum".

Attempts to specify more rewrite rules than allowed by this option's maximum (normally the automatic resizing maximum, unless a higher maximum has been explicitly configured) will result in an mm_init error, "no room in channel host table for ...". In particular, note the following. In its literal meaning, the "no room in channel host table for ..." error indicates that your configuration's current MTA internal table sizes are not large enough for the number of host names listed in your channel definitions. However, note that an extraneous blank line in the rewrite rules (upper portion) of your MTA configuration file causes the MTA to interpret the remainder of the configuration file as channel definitions; with just one such extraneous blank line, the MTA sees just one extra channel but with a lot of (all the rest of the rewrite rules as) host names on that channel. So check the line of the file that the error is complaining about: if it is not truly intended as a host name on a channel definition but rather is a line in the rewrite rules section of your configuration file, then check for an extraneous blank line somewhere *above* it.

39.22.14 Internal size MTA options: ldap_attr_name_hash_size (1-1000000)

The ldap_attr_name_hash_size MTA option specifies the size of the internal table of LDAP attribute names. The default is 64; the maximum value allowed is 1,000,000. Attempts to set this option's value higher than the maximum of 1,000,000 will result in an mm_init error, "LDAP_ATTR_NAME_HASH_SIZE exceeds maximum". Attempts to use more LDAP attributes than this option's maximum will result in an mm_init error, "Unable to expand LDAP attribute name hash table". If a system does not have enough memory to allocate the space required by this option's value, then an mm_init error will result, "Unable to allocate LDAP attribute name hash array".

39.22.15 Internal size MTA options: ldap_object_class_hash_size (1-1000000)

The ldap_object_class_hash_size MTA option controls the size of the internal table of LDAP object classes known to the MTA. The default is 64; the maximum is 1,000,000. Attempts to set this option to more than its maximum value of 1,000,000 will result in an mm_init error, "LDAP_OBJECT_CLASS_HASH_SIZE exceeds maximum". Attempts to specify more LDAP

object classes than are permitted by this option's maximum will result in an mm_init error, "Unable to expand LDAP object class hash table". If a system has insufficient memory to allocate the space required by this option's value, then an mm_init error will result, "Unable to allocate LDAP object class hash array".

39.22.16 Internal size MTA options: map_names_size (1-1000000)

The map_names_size option specifies the size of the mapping table name table, and thus the total number of [mapping tables](#) cannot exceed this number. The default is 32; the maximum allowed with normal, automatic resizing is 5000 (controlled from the maximum.dat file); the "hard" maximum is 1,000,000. Attempts to set this option higher than its maximum of 1,000,000 will result in an mm_init error, "MAP_NAMES_SIZE exceeds maximum".

Attempts to specify more mapping tables than allowed by this option's maximum (normally the automatic resizing maximum, unless a higher maximum has been explicitly configured) will result in an mm_init error, "no room in table for mapping named ...". In particular, note the following. In its literal meaning, the "no room in table for mapping named ..." error indicates that your configuration's current MTA internal table sizes are not large enough for your current number of mapping tables. However, also note that formatting errors in the MTA mapping file can cause the MTA to think that you have more mapping tables than you really have; for instance, check that mapping table entries are all properly indented.

39.22.17 Internal size MTA options: options_hash_size (1-1000000)

The options_hash_size MTA option sets the internal hash size for MTA options (Unified Configuration mta.option-name options or legacy configuration option.dat options). The default is 512; the maximum is 1,000,000. Attempts to set this option to more than its maximum 1,000,000 will result in an mm_init error, "OPTIONS_HASH_SIZE exceeds maximum".

39.22.18 Internal size MTA options: personal_conversion_size(1-2000)

RESTRICTED. The "personal" conversions feature is not expected to be used nowadays.

The personal_conversion_size MTA option sets the number of "personal" conversion entries each login user is permitted to have in their personal conversions file. The default is 32; the maximum to which this option may be set is 2000.

39.22.19 Internal size MTA options: reverse_data_size (1-1000000)

The reverse_data_size MTA option controls the internal hash size for the reverse (database) text file entries. The default is 256; the maximum is 1,000,000. Attempts to set this option's value to higher than its maximum of 1,000,000 will result in an mm_init error, "REVERSE_DATA_SIZE exceeds maximum". Attempts to specify more entries in the reverse

(database) text file than this option's maximum will result in an mm_init error, "no room in table for reverse data ...".

39.22.20 Internal size MTA options: string_pool_size_n (1-5000000)

The string_pool_size_n MTA options, string_pool_size_0,... string_pool_size_4, were introduced in JES MS 6.0; (formerly, the single option string_pool_size controlled the total number of strings available to the MTA for general configuration use). The string_pool_size_n options control the number of character slots allocated to the string pools used to hold rewrite rule templates, alias list members, mapping entries, etc. A fatal error will occur if the total number of characters consumed by these parts of the configuration files exceed these limits. The default for each of these options is 32,000; the maximum allowed value for each of these options is 50,000,000 as of JES MS 6.3; (10,000,000 in JES MS 6.2 and earlier). Attempts to set any of these options' values higher than their respective maximums (of 10,000,000 each in earlier versions, or 50,000,000 each as of JES MS 6.3) will result in the maximum value (of 10,000,000, or 50,000,000 as of JES MS 6.3) being used, with no error message.

string_pool_size_0 controls the string pool size for miscellaneous strings, including rewrite rule templates, [MTA option values](#), [channel option values](#), and strings in MTA [conversions entries](#). string_pool_size_1 controls the string pool size for [mapping tables](#). string_pool_size_2 controls the string pool size for [aliases](#). string_pool_size_3 controls the string pool size for the templates (right hand sides) of [database text files](#), including the [general \(database\) text file](#), the reverse (database) text file, and the forward (database) text file. string_pool_size_4 controls the string pool size for personal conversions.

39.22.21 Internal size MTA options: wild_pool_size (1-20000)

The wild_pool_size MTA option controls the total number of patterns that may appear throughout mapping tables. A fatal error will occur if the total number of mapping patterns exceeds this limit. The default is 8,000; the maximum allowed with normal, automatic resizing is 100,000 (controlled from the maximum.dat file); the "hard" maximum allowed value is 200,000. Attempts to set this option's value higher than its maximum of 200,000 will result in a value of 200,000 being used, with no error message.

39.23 LDAP external directory lookup MTA options

There are options controlling the LDAP fundamentals of any [extldap:url lookups](#) the MTA any be configured to perform. Such lookups, hence such options, were added for Messaging Server 7.2-7.02.

While the feature, new in Messaging Server 7.4-18.01, to allow explicit host and port specifications in LDAP URLs configured for MTA use means that these options might not all seem quite as necessary -- in operation since an external LDAP directory will likely require different bind credentials than an internal LDAP directory, as well as potentially other different settings, it is generally still useful to specifically configure using extldap: lookups instead of ldap: lookups.

Example uses for LDAP external directory lookups -- LDAP lookups directed against some LDAP directory other than the one used for regular user/group data -- would be cases where [Sieve external lists](#) are stored in some external LDAP directory, or for cases of looking up some data in [mapping tables from an LDAP directory](#) other than the usual LDAP directory.

39.23.1 LDAP external directory lookup MTA options: `ldap_ext_host` (host-name)

The `ldap_ext_host` MTA option takes a host name argument, and specifies the host to which to connect when making external LDAP (`extldap:`) queries. There is no default; this option must be explicitly set in order to successfully perform `extldap:` queries.

39.23.2 LDAP external directory lookup MTA options: `ldap_ext_max_connections` (non-negative integer)

The `ldap_ext_max_connections` MTA option takes a non-negative integer argument specifying the maximum number of simultaneous connections to permit to the external LDAP server. The default is 1024.

39.23.3 LDAP external directory lookup MTA options: `ldap_ext_password` (string)

The `ldap_ext_password` MTA option takes a string argument as the password used to authenticate for LDAP external directory (`extldap:`) queries. There is no default.

39.23.4 LDAP external directory lookup MTA options: `ldap_ext_port` (port)

The `ldap_ext_port` MTA option takes a non-negative integer argument specifying the port for the LDAP external directory. If not set, this defaults to the value of the [ldap_port](#) MTA option (which itself defaults to 389, the standard LDAP server port number).

39.23.5 LDAP external directory lookup MTA options: `ldap_ext_username` (DN)

The `ldap_ext_username` MTA option takes a DN specifying the username (bind credentials) for LDAP external directory (`extldap:`) queries. There is no default value: if no DN is specified, then authentication will not be used.

39.24 LDAP PAB MTA options

There are MTA options controlling the LDAP fundamentals of any personal addressbook (PAB) lookups the MTA may be configured to perform; that is, options affecting the MTA's handling of [pabldap:url lookups](#). Such lookups, hence such options, were added for Messaging Server 7.0-0.04, though not all the functionality was fully fleshed out for that version.

Note that in order for the MTA to successfully perform any such PAB lookups, typically it is necessary to add an ACI to the PAB to allow the MTA read access. Such an ACI might be along the lines of:

```
dn: o=piServerDb
changetype: modify
add: aci
aci: (target="ldap:///o=piServerDb")
    (targetattr="*")
    (version 3.0; acl "PAB Administrator read rights"; allow (read,search)
    groupdn="ldap:///cn=Messaging End User Administrator Group, ou=groups, o=isp";
```

39.24.1 LDAP PAB MTA options: `ldap_pab_host` (hostname)

(New in 7.0-0.04) The `ldap_pab_host` MTA option specifies the host for the MTA's PAB LDAP queries; overrides for MTA PAB lookup purposes the [ldaphostPAB option](#) (`local.service.pab.ldaphost` configutil parameter in legacy configuration).

39.24.2 LDAP PAB MTA options: `ldap_pab_max_connections` (integer)

(New in Messaging Server 7.0u1) The `ldap_pab_max_connections` MTA option specifies a limit on the maximum number of connections that will be made by the MTA to the PAB Directory Server. The default is 1024.

39.24.3 LDAP PAB MTA options: `ldap_pab_password` (string)

(New in 7.0-0.04) The `ldap_pab_password` MTA option specifies the password for the MTA's PAB LDAP queries; overrides for MTA PAB lookup purposes the [ldappasswdPAB option](#) (`local.service.pab.ldappasswd` configutil parameter in legacy configuration).

39.24.4 LDAP PAB MTA options: `ldap_pab_port` (integer)

(New in 7.0-0.04) The `ldap_pab_port` MTA option specifies the port for the MTA's PAB LDAP queries; overrides for MTA PAB lookup purposes the [ldappportPAB option](#) (`local.service.pab.ldappport` configutil parameter in legacy configuration).

39.24.5 LDAP PAB MTA options: `ldap_pab_username` (dn)

(New in 7.0-0.04) The `ldap_pab_username` MTA option specifies the username (bind credentials) for the MTA's PAB LDAP queries; overrides for MTA PAB lookup purposes the [ldapbinddnPAB option](#) (`local.service.pab.ldapbinddn` configutil parameter in legacy configuration).

39.25 Mailing list MTA options

The MTA has a number of options relating specifically to mailing lists and groups.

As mailing lists and groups are merely a special form of alias, see also the [general MTA options relating to aliases](#). And for mailing lists or groups defined in LDAP, see also the [Direct LDAP MTA options](#), and specifically the [Direct LDAP usergroup lookup MTA options](#).

39.25.1 Mailing list controls (`defer_group_processing`)

The `defer_group_processing` MTA [mailing list option](#) sets a default for whether [direct LDAP group \(list\) expansion](#) is performed "in-line", or whether it is deferred (performed via the [reprocess channel](#)). The default is 1 (true); that is, by default group expansion is deferred to the reprocess channel.

Note that when an address is deferred, the usual incoming [recipient access mapping table](#) (specifically, [ORIG_SEND_ACCESS](#), [SEND_ACCESS](#), [ORIG_MAIL_ACCESS](#), and [MAIL_ACCESS](#)) checks are not performed; such checks don't take place until the reprocess channel runs and expands the address. Also, the originator address checking (via attributes such as *mgrpAllowedBroadcaster*, etc.) is deferred. (If the [dis]allowed broadcasters are a large dynamic group themselves, deferring this expansion is very desirable so that that lookup doesn't slow down the SMTP dialogue.) The [reprocess channel](#) when it runs will then perform all the normal access checks as if it were the original channel. In particular, the original transport and application fields (used in [ORIG_MAIL_ACCESS](#) and [MAIL_ACCESS](#) mapping tables) are available to the reprocess channel, and hence checks such as [dns_verify callouts](#) can still be performed at this later time.

If `defer_group_processing=0` is set, so that the default is that groups are expanded "in-line" (for instance, during an SMTP dialogue), then deferred expansion may still be set for particular groups either by setting the [mailDeferProcessing attribute](#) (or more precisely whatever attribute is named by the [ldap_reprocess MTA option](#)) to *Yes*, or (deprecated) by setting the (multi-valued) [mailDeliveryOption attribute](#) (or more precisely, whatever attribute is named by the [ldap_delivery_option MTA option](#)) to have a value `members_offline`.

39.25.2 Mailing list and group MTA options: `digest_on` (string)

The `digest_on` MTA [mailing list option](#) specifies the comment string that enables [mailing list digests](#) (in preference to regular mailing list postings). The default is "(DIGEST)".

Caution: Usage of this option is **RESTRICTED**. It has not yet been fully implemented. Do not use this option unless and until instructed to do so by Oracle.

See also: [Mailing lists](#).

39.25.3 Mailing list and group MTA options: `expandable_default` (0 or 1)

The `expandable_default` MTA option establishes a default for whether lists are expandable; in particular, it establishes a default for whether the EXPN SMTP command may be used to expand lists (display list membership).

The default value for this option is 1, meaning that by default lists are expandable. Note that this default may be overridden on a per list basis. For aliases in Unified Configuration, the `alias_expandable` and `alias_nonexpandable` alias options override whatever default is set via the `expandable_default` MTA option. For lists defined in the `aliases file` or `alias database`, the `[EXPANDABLE]` or `[NONEXPANDABLE]` named parameters override whatever default is set via the `expandable_default` MTA option. For lists defined purely in LDAP, the `mgmanMemberVisibility` and `expandable` attributes (more precisely, those attributes named by the `ldap_expandable` MTA option) override the default set via `expandable_default`.

Also note that for mailing lists with posting access controls, such posting access controls affect when expansion via the SMTP EXPN command is allowed; the SMTP server will only permit an EXPN if the SMTP client passes the posting access control (*e.g.*, has issued a prior MAIL FROM: command that passes the access control). Note also that the TCP/IP-channel-specific option `DISABLE_EXPAND` may be used to disable the EXPN SMTP command entirely for those incoming TCP/IP channels corresponding to that default TCP/IP channel's SMTP server; see [TCPIP-channel-specific options](#).

39.25.4 Mailing list and group MTA options: `mail_off` (string)

The `mail_off` MTA option specifies the comment string, including the surrounding parentheses, that disables mail delivery for list addresses specified in the `alias file` or `alias database`, or set via an `alias setting in Unified Configuration`. The default value for this string is `(NOMAIL)`.

39.25.5 Mailing list and group MTA options: `or_clauses` (0 or 1)

The `or_clauses` MTA option sets the default for whether multiple mailing list access control settings (*e.g.*, `alias_auth_list`, `alias_cant_list`, *etc.*, or `[AUTH_LIST]`, `[CANT_LIST]`, *etc.* in legacy configuration) are ANDed (the default, `or_clauses=0`) or ORed (`or_clauses=1`). See [Mailing list multiple access control interpretation](#) or additional discussion.

This general MTA option setting can be overridden on a per list or group basis via the alias options `alias_or` or `alias_and` (in legacy configuration, the mailing list named parameters `[OR]` or `[AND]`), or in direct LDAP mode via use of an "OR" or "AND" value as one of the values of the LDAP attribute named by the `ldap_auth_policy` MTA option (by default, `mgrpBroadcasterPolicy`).

39.25.6 Mailing list and group MTA options: `post_off` (string)

The `post_off` MTA option specifies the comment string, including the surrounding parentheses, that disables mail posting for list addresses specified in the `alias file` or `alias database`, or set via an `alias setting in Unified Configuration`. The default is `(NOPOST)`.

39.26 MAILSERV MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

39.26.1 MAILSERV moderator MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

When using MAILSERV, a special MAILSERV moderator user must be set up, that will handle certain MAILSERV functions. MTA options exist to specify this moderator user.

39.26.1.1 MAILSERV moderator user MTA options: `mailserv_moderator_mail` (RFC 822 address)

RESTRICTED. Not yet fully implemented.

The `mailserv_moderator_mail` MTA option specifies the e-mail address of the MAILSERV "user": the account whose Message Store mailbox is used by MAILSERV for list moderation functions.

39.26.1.2 MAILSERV moderator user MTA options: `mailserv_moderator_uid` (uid)

RESTRICTED. Not yet fully implemented.

The `mailserv_moderator_uid` MTA option specifies the uid of the MAILSERV "user": the account whose Message Store mailbox is used by MAILSERV for list moderation functions.

Note that the value must be a valid uid value; in particular, only a subset of ASCII characters are permitted; see the [ldap_uid_invalid_chars MTA option](#).

39.26.1.3 MAILSERV moderator MTA options: `mailserv_secret` (string)

RESTRICTED. Not yet fully implemented.

The `mailserv_secret` MTA option is used internally to generate passwords used internally for lists managed by MAILSERV. It has no default.

39.26.2 MAILSERV LDAP schema MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

There are a couple of MTA options specifying basics of the LDAP schema for MAILSERV.

39.26.3 MAILSERV user LDAP attribute name MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

By default, the MTA assumes a particular sort of LDAP schema will be used with MAILSERV; in particular, the MTA assumes that certain named attributes (with certain sorts of meanings) are available and used in the LDAP directory to store MAILSERV user data, and MAILSERV list subscription data. However, the exact attribute names that the MTA looks for (recognizes) are configurable via the various `ldap_mluser_*` and `ldap_mlsub_*` MTA options. Thus a different (though semantically compatible) schema may be used by setting the `ldap_mluser_*` and `ldap_mlsub_*` MTA options to tell the MTA what named attributes to use (recognize). In addition to the MTA options listed here, see also the `ldap_mluser_object_class` MTA option which specifies the object class(es) for MAILSERV users.

Note that throughout MAILSERV discussions, for convenience often LDAP attributes will be referred to merely by name. But in general, any such MAILSERV reference to a specific LDAP attribute name really ought to be a reference to the attribute named by the corresponding MTA option. For instance, any use by MAILSERV of the `mail` LDAP attribute is really a use of the attribute named by the `ldap_mluser_mail` MTA option.

39.26.4 MAILSERV list subscription LDAP attribute name MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

By default, the MTA assumes a particular sort of LDAP schema will be used with MAILSERV; in particular, the MTA assumes that certain named attributes (with certain sorts of meanings) are available and used in the LDAP directory to store MAILSERV user data, and MAILSERV list subscription data. However, the exact attribute names that the MTA looks for (recognizes) are configurable via the various `ldap_mluser_*` and `ldap_mlsub_*` MTA options. Thus a different (though semantically compatible) schema may be used by setting the `ldap_mluser_*` and `ldap_mlsub_*` MTA options to tell the MTA what named attributes to use (recognize). In addition to the MTA options listed here, see also the `ldap_mlsub_object_class` MTA option which specifies the object class for MAILSERV list subscriptions.

Note that throughout MAILSERV discussions, for convenience often LDAP attributes will be referred to merely by name. But in general, any such MAILSERV reference to a specific LDAP attribute name really ought to be a reference to the attribute named by the corresponding MTA option. For instance, any use by MAILSERV of the `mlsubListIdentifier` LDAP attribute is really a use of the attribute named by the `ldap_mlsub_list_id` MTA option.

39.26.5 MAILSERV list LDAP attribute name MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

By default, the MTA assumes a particular sort of LDAP schema will be used with MAILSERV; in particular, the MTA assumes that certain named attributes (with certain sorts of meanings) are available and used in the LDAP directory to store mailing list data. However, the exact attribute names that the MTA looks for (recognizes) on list entries managed by MAILSERV are configurable via the various `ldap_list_*` MTA options. Thus a different (though semantically compatible) schema may be used by setting the `ldap_list_*` MTA options to tell the MTA what named attributes to use (recognize).

Note that throughout MAILSERV discussions, for convenience often LDAP attributes will be referred to merely by name. But in general, any such MAILSERV reference to a specific LDAP

attribute name really ought to be a reference to the attribute named by the corresponding MTA option. For instance, any use by MAILSERV of the `memberOf` LDAP attribute is really a use of the attribute named by the [ldap_list_name MTA option](#).

39.27 Mapping table MTA options

A number of MTA options affect the syntax or operation of MTA [mapping tables](#). For such options affecting in particular the access mapping tables, see [Access mapping table MTA options](#); for options affecting MTA mapping table syntax in general, or the use of other (non-access) MTA mapping tables, see [Miscellaneous mapping table MTA options](#).

39.27.1 Access mapping table MTA options

This discussion will focus on those MTA options that specifically or primarily affect the [*_ACCESS mapping tables](#) as well as mailing list access mapping tables. In particular, many of these options alter the format of [*_ACCESS](#) mapping table probes by causing inclusion of additional fields, thus additional information, in the [*_ACCESS](#) mapping table probes. The [use_auth_return](#), [use_canonical_return](#), and [use_orig_return](#) options can, among other things, affect the form of envelope From address used in [recipient-address-based *_ACCESS mapping table](#) probes and in mailing list named parameter [`*_MAPPING`] mapping table probes. See also the [access_errors](#) MTA option for control of the default error text resulting from recipient address [*_ACCESS](#) mapping table rejections. And see also [Miscellaneous mapping table MTA options](#) for a discussion of additional options with more miscellaneous (less specifically focused on access mapping table) effects.

39.27.1.1 Access mapping table MTA options: `access_auth` (bitmask)

The `access_auth` MTA option is used to cause inclusion of additional authentication information in mapping probes. This is a bit-encoded field; currently two bits are defined. If bit 0 (value 1) is set, the value of the SMTP AUTH parameter is included in the [FROM_ACCESS mapping](#) probe immediately following the authenticated sender address. If bit 1 (value 2) is set, the canonical username result produced by authentication is included in the [FROM_ACCESS](#) mapping probe immediately following the SMTP AUTH parameter. The default value for this option is 0.

39.27.1.2 Access mapping table MTA options: `access_counts` (bitmask)

(New in JES MS 6.3.) The `access_counts` MTA option provides a way to get at various types of recipient count information in the various [recipient-based *_ACCESS mappings](#). `access_counts` is bit-encoded in the same way as [access_orcpt](#) is, and if set, enables the addition of a set of counts after the (optional) `access_orcpt` field and before the (optional) [include_conversiontag](#) field in the access mapping probe string. Currently the format of the count addition is:

```
RCPT-TO-count/total-recipient-count/
```

Note the trailing slash. It is expected that additional counter information will be added to this field in the future; all mappings making use of this information should be coded to ignore anything following the (current) last slash or they may break without warning. The

`total-recipient-count` is the count of valid recipients resulting from previous RCPT TO commands (*not* including the RCPT TO command corresponding to this probe).

Table 39.20 `access_counts` MTA option bit values

Bit	Value	Usage
0	1	If set, include recipient counts in each <code>[ORIG_]SEND_ACCESS</code> and <code>[ORIG_]MAIL_ACCESS</code> mapping table probe
1	2	If set, include recipient counts in each <code>ORIG_SEND_ACCESS</code> probe
2	4	If set, include recipient counts in each <code>SEND_ACCESS</code> probe
3	8	If set, include recipient counts in each <code>ORIG_MAIL_ACCESS</code> probe
4	16	If set, include recipient counts in each <code>MAIL_ACCESS</code> probe

Bit 0 is the least significant bit.

39.27.1.3 Access mapping table MTA options: `access_orcpt` (bitmask)

The `access_orcpt` option's default value is 0. Setting `access_orcpt` to 1 adds an additional vertical bar delimited field to the probe value passed to the [recipient access mapping tables](#) (that is, the `SEND_ACCESS`, `ORIG_SEND_ACCESS`, `MAIL_ACCESS`, and `ORIG_MAIL_ACCESS` mapping tables) that contains the original recipient (ORCPT) address (or if the message doesn't have an ORCPT address, then the original unmodified RCPT TO: address is used instead). A new feature added for JES MS 6.3 are separate bits to separately control inclusion of the ORCPT value in these various mapping table probes. Note that such values take the form of an address type string, followed by a semicolon, followed by the address. For instance, `rfc822;user@domain.com`.

Table 39.21 `access_orcpt` MTA option bit values

Bit	Value	Usage
0	1	If set, include ORCPT value in each <code>[ORIG_]SEND_ACCESS</code> and <code>[ORIG_]MAIL_ACCESS</code> mapping table probe
1	2	(New in 6.3) If set, include ORCPT value in each <code>ORIG_SEND_ACCESS</code> probe
2	4	(New in 6.3) If set, include ORCPT value in each <code>SEND_ACCESS</code> probe
3	8	(New in 6.3) If set, include ORCPT value in each <code>ORIG_MAIL_ACCESS</code> probe
4	16	(New in 6.3) If set, include ORCPT value in each <code>MAIL_ACCESS</code> probe

Bit 0 is the least significant bit.

39.27.1.4 Access mapping table MTA options: `include_connectioninfo` (bitmask)

(New in JES MS 6.2.) This option selectively enables the inclusion of the transport and application connection information in various [mapping table](#) probes that otherwise would not include this material. The relevant mapping tables correspond to certain [alias options](#) in Unified Configuration, or in legacy configuration to certain [named \(that is, nonpositional\)](#)

[alias parameters in the alias file](#), most often used on mailing lists defined via the alias file. If included, the connection information appears at the beginning of the mapping probe in the same format used in the [FROM_ACCESS](#), [MAIL_ACCESS](#), and [ORIG_MAIL_ACCESS](#) mappings. The option takes a bit-encoded value that defaults to 0. The following bits are defined:

Table 39.22 include_connectioninfo MTA option bit values

Bit	Value	Usage
0	1	AUTH_MAPPING or alias_auth_mapping
1	2	MODERATOR_MAPPING or alias_moderator_mapping
2	4	CANT_MAPPING or alias_cant_mapping
3	8	DEFERRED_MAPPING or alias_deferred_mapping
4	16	DIRECT_MAPPING or alias_direct_mapping
5	32	HOLD_MAPPING or alias_hold_mapping
6	64	NOHOLD_MAPPING or alias_nohold_mapping
7	128	SASL_AUTH_MAPPING or alias_sasl_auth_mapping
8	256	SASL_MODERATOR_MAPPING or alias_sasl_moderator_mapping
9	512	SASL_CANT_MAPPING or alias_sasl_cant_mapping

Bit 0 is the least significant bit.

39.27.1.5 Access mapping table MTA options: include_conversiontag (bitmask)

New in JES MS 6.3. This option selectively enables the inclusion of [conversion tag](#) information in various mapping table probes. When enabled, the current set of conversion tags will appear in the relevant mapping table probe as a comma separated list. The option takes a bit-encoded integer value. The following bits are defined:

Table 39.23 include_conversiontag MTA option bit values

Bit	Value	Usage
0	1	CHARSET-CONVERSION : include as a ;TAG=comma-separated-values field before ;CONVERT
1	2	CONVERSIONS : include as a ;TAG=comma-separated-values field before ;CONVERT
2	4	FORWARD : include just before the current recipient address (with a vertical bar, , delimiter)
3	8	ORIG_SEND_ACCESS : include as a final (barring any ldap_spare_* fields) field; that is, include after the optional access count field and prior to any ldap_spare_* fields (with a vertical bar, , delimiter)
4	16	SEND_ACCESS : include as a final (barring any ldap_spare_* fields) field; that is, include as a field after the optional access count field and prior to any ldap_spare_* fields (with a vertical bar, , delimiter)

5	32	ORIG_MAIL_ACCESS : include as a final (barring any <code>ldap_spare_*</code> fields) field; that is, include as a field after the optional access count field and prior to any <code>ldap_spare_*</code> fields (with a vertical bar, <code> </code> , delimiter)
6	64	MAIL_ACCESS : include as a final (barring any <code>ldap_spare_*</code> fields) field; that is, include as a field after the optional access count field and prior to any <code>ldap_spare_*</code> fields (with a vertical bar, <code> </code> , delimiter)
7	128	FROM_ACCESS : (New in 7.0-3.01? post JES MS 6.3) include as a final (barring any <code>ldap_spare_*</code> fields) field; that is, include as a field after the optional access count field and prior to any <code>ldap_spare_*</code> fields (with a vertical bar, <code> </code> , delimiter)
8	256	REVERSE : (New in 7.0.5) include as a field after any source or destination channel fields, but before the actual address in the probe (with a vertical bar, <code> </code> , delimiter). Note that only the "final" REVERSE mapping table probe, used to (possibly) modify addresses as a message is being enqueued, includes the conversion tags; earlier probes, such as for (possibly) modifying the envelope From for <code>*_ACCESS</code> mapping table probe purposes, or for (possibly) setting an "override" postmaster address do not include conversion tags regardless of the setting of this bit of <code>include_conversiontag</code> .
9	512	PERSONAL_NAMES : (New in 8.0) include as a field after any source or destination channel fields, but before any addresses in the probe (with a vertical bar, <code> </code> , delimiter).
10	1024	Domain catchall mappings : (New in 8.0) include as a field after any authenticated sender address , but before any MT-PRIORITY and recipient address in the probe (with a vertical bar, <code> </code> , delimiter).

Bit 0 is the least significant bit.

The default is 0, meaning that conversion tag values are not included in any mapping table probes.

39.27.1.6 Mapping table MTA options: `include_mtpriority` (bitmask)

(New in 8.0.) The current MT-PRIORITY value (an integer in the range -9 to 9) and the current message size estimate may be included in various [mapping](#) probes. This is all controlled by the `include_mtpriority` MTA option. When these values are included in a probe they appear as separate fields. This option is bit-encoded, with the bits defined as follows:

Table 39.24 `include_mtpriority` MTA option values

Bit	Value	Meaning
0	1	Include MT-PRIORITY and size estimate in FROM_ACCESS probes immediately after any include_spares values
1	2	Include MT-PRIORITY and message size estimate in FORWARD probes immediately after any conversion tag field (resulting from setting the include_conversiontag MTA option)

Access mapping table MTA options

2	4	Include MT-PRIORITY and message size estimate in ORIG_SEND_ACCESS probes immediately after any include_spares values
3	8	Include MT-PRIORITY and message size estimate in SEND_ACCESS probes immediately after any include_spares values
4	16	Include MT-PRIORITY and message size estimate in ORIG_MAIL_ACCESS probes immediately after any include_spares values
5	32	Include MT-PRIORITY and message size estimate in MAIL_ACCESS probes immediately after any include_spares values
6	64	Append the MT-PRIORITY and message size estimate values in the form ";MT-PRIORITY=<value>;BLOCKS=<value>" to the CONVERSIONS mapping probe immediately after any "TAG=" clause.
7	128	Append the MT-PRIORITY and message size estimate values in the form ";MT-PRIORITY=<value>;BLOCKS=<value>" to any domain catchall mapping probe immediately after any conversion tag clause and before the recipient address.

Bit 0 is the least significant bit.

The default is 0, meaning that no MT-PRIORITY or message size estimates are included in the various mapping table probes. With the exception of the [CONVERSIONS](#) mapping, the MT-PRIORITY and message size estimate are added in that order, delimited by vertical bars.

The message size estimate is the size of the queued message entry for internal channels; it's the value given by the SMTP SIZE extension for incoming SMTP channels. The size is given in [MTA blocks](#) and will be 0 if no size information is available. Note that message sizes can change as a result of channel processing, encoding, decoding, and conversion operations.

39.27.1.7 Access mapping table MTA options: [include_spares](#) (bitmask)

(New in Messaging Server 7u2.) LDAP attribute values associated with originator or recipient address processing may be included in [FROM_ACCESS](#) or the various [recipient address access mapping probes](#), respectively. This is all controlled by the [include_spares](#) MTA option. This option is bit-encoded, with the bits defined as follows:

Table 39.25 [include_spares](#) MTA option values

Bit	Value	Meaning
0	1	Include sender ldap_spares_1 attribute in FROM_ACCESS probes
1	2	Include sender ldap_spares_2 attribute in FROM_ACCESS probes
2	4	Include sender ldap_spares_3 attribute in FROM_ACCESS probes
3	8	Include sender ldap_spares_4 attribute in FROM_ACCESS probes
4	16	Include sender ldap_spares_5 attribute in FROM_ACCESS probes
5	32	Include sender ldap_spares_6 attribute in FROM_ACCESS probes
6	64	Include recipient ldap_spares_1 attribute in ORIG_SEND_ACCESS probes
7	128	Include recipient ldap_spares_2 attribute in ORIG_SEND_ACCESS probes

8	256	Include recipient <code>ldap_spare_3</code> attribute in <code>ORIG_SEND_ACCESS</code> probes
9	512	Include recipient <code>ldap_spare_4</code> attribute in <code>ORIG_SEND_ACCESS</code> probes
10	1024	Include recipient <code>ldap_spare_5</code> attribute in <code>ORIG_SEND_ACCESS</code> probes
11	2048	Include recipient <code>ldap_spare_6</code> attribute in <code>ORIG_SEND_ACCESS</code> probes
12	4096	Include recipient <code>ldap_spare_1</code> attribute in <code>SEND_ACCESS</code> probes
13	8192	Include recipient <code>ldap_spare_2</code> attribute in <code>SEND_ACCESS</code> probes
14	16384	Include recipient <code>ldap_spare_3</code> attribute in <code>SEND_ACCESS</code> probes
15	32768	Include recipient <code>ldap_spare_4</code> attribute in <code>SEND_ACCESS</code> probes
16	65536	Include recipient <code>ldap_spare_5</code> attribute in <code>SEND_ACCESS</code> probes
17	131072	Include recipient <code>ldap_spare_6</code> attribute in <code>SEND_ACCESS</code> probes
18	262144	Include recipient <code>ldap_spare_1</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
19	524288	Include recipient <code>ldap_spare_2</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
20	1048576	Include recipient <code>ldap_spare_3</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
21	2097152	Include recipient <code>ldap_spare_4</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
22	4194304	Include recipient <code>ldap_spare_5</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
23	8388608	Include recipient <code>ldap_spare_6</code> attribute in <code>ORIG_MAIL_ACCESS</code> probes
24	16777216	Include recipient <code>ldap_spare_1</code> attribute in <code>MAIL_ACCESS</code> probes
25	33554432	Include recipient <code>ldap_spare_2</code> attribute in <code>MAIL_ACCESS</code> probes
26	67108864	Include recipient <code>ldap_spare_3</code> attribute in <code>MAIL_ACCESS</code> probes
27	134217728	Include recipient <code>ldap_spare_4</code> attribute in <code>MAIL_ACCESS</code> probes
28	268435456	Include recipient <code>ldap_spare_5</code> attribute in <code>MAIL_ACCESS</code> probes
29	536870912	Include recipient <code>ldap_spare_6</code> attribute in <code>MAIL_ACCESS</code> probes

Bit 0 is the least significant bit.

The default is 0, meaning that no LDAP spare attribute values are included in the `*_ACCESS` mapping table probes. If inclusion of any spare attribute values is enabled, those spare attribute values are suffixed to the probe after the optional (see `include_conversiontag`) conversion tag value(s). Any spare attribute values enabled are suffixed in order (first a vertical bar and `ldap_spare_1` if enabled, then a vertical bar and `ldap_spare_2` if enabled, etc.).

Note that spare attribute slots can be assigned to point to attributes already used for other purposes by the MTA. That is, via this mechanism, the `*_ACCESS` mapping tables can in fact be sensitive to the value of any attribute available to the MTA during the relevant mapping table probe: just about any recipient or sender attribute likely to be of interest could be made available to the mapping probe.

39.27.1.8 Miscellaneous mapping table MTA options: `include_spare1` (bitmask)

(New in 8.0.) Replaces the former `include_spare` MTA option.

For a similar feature of adding LDAP attribute values to `FORWARD mapping table` probes, see also the `include_spare2` MTA option.

39.27.1.9 Access mapping table MTA options: `mapping_paranoia` (integer)

(New in Messaging Server 7.0) Since access-check mappings such as the [recipient address * _ACCESS mappings](#), the [FROM_ACCESS mapping](#), the [AUTH_REWRITE mapping](#), the [BURL_ACCESS mapping](#), the [SIEVE_EXTLISTS mapping](#), and new in Messaging Server 7.3-11.01 the [MILTER_MACROS mapping table](#), and new in 7.0.5 the [GROUP_AUTH mapping table](#) and [AUTH_ACCESS mapping table](#), use vertical bars as delimiters, issues can arise when externally provided material such as envelope From or To addresses themselves contain vertical bars. The `mapping_paranoia` MTA option is intended to provide various tools to handle such issues.

Giving the `mapping_paranoia` MTA option a nonnegative value will cause any vertical bars in the externally supplied portions of various mapping input strings to be replaced in the mapping probe with the character whose ASCII value is given by this option. (Attempting to set `mapping_paranoia` to a positive value greater than 127 will result in the value 124, the default, being used.) That is, the "regular", field-separating, vertical bars will still be present, but a vertical bar within an external field (such as within an address) will be replaced by the specified character. A negative value will cause the vertical bar to simply be dropped entirely from the probe. The default value for this option is 124, the ASCII value for vertical bar, which causes vertical bars to be left untouched.

Note that many mapping tables where `mappings_paranoia` is relevant also have a feature for testing whether a vertical bar was originally present in externally supplied probe fields; use of `mapping_paranoia` to replace the original vertical bar characters does not affect such testing: the test flag is set based on the original presence of a vertical bar character, regardless of whether it is later replaced due to `mapping_paranoia`.

39.27.1.10 Access mapping table MTA options: `use_ip_access` (bitmask)

New in Messaging Server 7.0. If bit 0 (value 1) of the `use_ip_access` MTA option is set, then a delivery attempt count field will be suffixed to the end of the [IP_ACCESS mapping table](#) probes. The default is 0.

39.27.1.11 Return address type used in checks MTA options: `use_auth_return`, `use_canonical_return`, `use_orig_return`

The MTA maintains three different forms of the envelope From address: The original from the MAIL FROM SMTP command (or its equivalent in other protocols), one that has had address reversal applied, and one that has been fully canonicalized, which includes mapping of `mailEquivalentAddress` attribute matches to the corresponding `mail` attribute. Additionally, there may be an address produced as a result of an authentication operation that has similar semantics.

There are many places where the MTA performs comparisons against or constructs mapping probes containing the "return" address. But since there are multiple "return" addresses, there needs to be a way to select the one that is used. This is controlled by the `use_auth_return`, `use_canonical_return`, and `use_orig_return` MTA options. Each of these options accepts a bit-encoded integer argument, with each bit controlling a particular place where a "return" address is used.

If the bit in `use_auth_return` is set and an authenticated address is available, it is used and the corresponding bits in the other options become no-ops. If the bit in `use_canonical_return` is set (and the one in `use_auth_return` is clear), then the canonicalized envelope From address is used and the corresponding bit in `use_orig_return` becomes a no-op. If the bit in `use_orig_return` is set (and the ones in the other two options are clear) then the original envelope from address is used. Finally, if none of the bits are set, the envelope From address that has had address reversal applied is used.

The `use_auth_return` MTA option was added in 7.0; `use_canonical_return` is first available in 6.3.

The uses of the return address the various bits control are described in the following table:

Table 39.26 use_auth_return MTA option bits

Bit	Value	Usage
0	1	ORIG_SEND_ACCESS mapping table probes
1	2	SEND_ACCESS mapping table probes
2	4	ORIG_MAIL_ACCESS mapping table probes
3	8	MAIL_ACCESS mapping table probes
4	16	Mailing list [AUTH_LIST], [MODERATOR_LIST], [SASL_AUTH_LIST], and [SASL_MODERATOR_LIST] checks, and in Unified Configuration alias_auth_list , alias_moderator_list , alias_sasl_auth_list , alias_sasl_moderator_list alias option checks
5	32	Mailing list [CANT_LIST] and [SASL_CANT_LIST] checks, and in Unified Configuration, alias_cant_list and alias_sasl_cant_list alias option checks
6	64	Mailing list [AUTH_MAPPING], [MODERATOR_MAPPING], [SASL_AUTH_MAPPING], and [SASL_MODERATOR_MAPPING] checks, and in Unified Configuration, alias_auth_mapping , alias_moderator_mapping , alias_sasl_auth_mapping , and alias_sasl_moderator_mapping alias option checks
7	128	Mailing list [CANT_MAPPING] and [SASL_CANT_MAPPING] checks, and in Unified Configuration, alias_cant_mapping and alias_sasl_cant_mapping alias option checks
8	256	Mailing list [ORIGINATOR_REPLY] comparisons, and in Unified Configuration, alias_originator_reply alias option comparisons
9	512	Mailing list [DEFERRED_LIST], [DIRECT_LIST], [HOLD_LIST], and [NOHOLD_LIST] checks, and in Unified Configuration, alias_deferred_list , alias_direct_list , alias_hold_list , and alias_nohold_list alias option checks
10	1024	Mailing list [DEFERRED_MAPPING], [DIRECT_MAPPING], [HOLD_MAPPING], and [NOHOLD_MAPPING] checks, and in Unified Configuration, alias_deferred_mapping , alias_direct_mapping , alias_hold_mapping , and alias_nohold_mapping alias option checks

11	2048	Mailing list checks for whether the sender is the list moderator
12	4096	Mailing list <code>ldap_auth_domain</code> LDAP attribute (e.g., <code>mgrpAllowedDomain</code>) checks
13	8192	Mailing list <code>ldap_cant_domain</code> LDAP attribute (e.g., <code>mgrpDisallowedDomain</code>) checks
14	16384	Mailing list <code>ldap_auth_url</code> LDAP attribute (e.g., <code>mgrpAllowedBroadcaster</code>) checks
15	32768	Mailing list <code>ldap_cant_url</code> LDAP attribute (e.g., <code>mgrpDisallowedBroadcaster</code>) checks
16	65536	OBSOLETE. In iMS 5.0 and 5.1 this controlled mailing list LDAP_MODERATOR_RFC822 comparisons; since as of iMS 5.2 there is no longer any such MTA option nor need for such an attribute (since the LDAP_MODERATOR_URL attribute value can, in fact, specify a mailto: URL pointing to an RFC 822 address), this bit no longer has any meaning.
17	131072	Mailing list <code>ldap_moderator_url</code> LDAP attribute (e.g., <code>mgrpModerator</code>) comparisons
18	262144	Source-specific FORWARD mapping table probes
19	524288	Source-specific FORWARD database probes
20	1048576	(New in 6.3) Source-specific domain catchall mappings
21	2097152	(New in 7.2-7.02) FROM_ACCESS mapping

The default for `use_auth_return` and `use_canonical_return` is 0. The default for `use_orig_return` is 0 prior to 7.2-7.02; in 7.2-7.02 and later it is 2097152, which preserves the existing default of using the original envelope from access in the FROM_ACCESS mapping. Also note that setting bit 21 in `use_auth_return` makes no sense as the authenticated sender address is available as a separate field in the probe.

39.27.2 Miscellaneous mapping table MTA options

This discussion lists MTA options affecting miscellaneous mapping tables. See also [Access mapping table MTA options](#) which lists those MTA options primarily affecting `*_ACCESS` and mailing list access mapping tables. And if performing explicit LDAP callouts in mapping tables, see also the `url_result_cache_*` MTA options controlling the per-process caching of results of such LDAP lookups.

39.27.2.1 averages_cache_size Option

Not currently implemented. The [MeterMaid](#) facility provides an alternate, more general, facility.

39.27.2.2 averages_cache_timeout Option

Not currently implemented. The [MeterMaid](#) facility provides an alternate, more general, facility.

39.27.2.3 Miscellaneous mapping table MTA options: include_spares2 (bitmask)

(New in 8.0.) LDAP attribute values associated with originator or recipient address processing may be included in [FORWARD](#) mapping probes. This is controlled by the `include_spares2` MTA option. This option is bit-encoded, with the bits defined as follows:

Table 39.27 `include_spares2` MTA option values

Bit	Value	Meaning
0	1	Include <code>ldap_spare_1</code> attribute in FORWARD probes
1	2	Include <code>ldap_spare_2</code> attribute in FORWARD probes
2	4	Include <code>ldap_spare_3</code> attribute in FORWARD probes
3	8	Include <code>ldap_spare_4</code> attribute in FORWARD probes
4	16	Include <code>ldap_spare_5</code> attribute in FORWARD probes
5	32	Include <code>ldap_spare_6</code> attribute in FORWARD probes

Bit 0 is the least significant bit.

The default is 0, meaning that no LDAP spare attribute values are included in [FORWARD mapping](#) probes. If inclusion of any spare attribute values is enabled, those spare attribute values are suffixed to the probe after the optional (see `include_conversiontag`) conversion tag value(s). Any spare attribute values enabled are suffixed in order (first a vertical bar and `ldap_spare_1` if enabled, then a vertical bar and `ldap_spare_2` if enabled, *etc.*).

Note that spare attribute slots can be assigned to point to attributes already used for other purposes by the MTA. That is, via this mechanism, the [FORWARD mapping table](#) can in fact be sensitive to the value of any attribute available to the MTA during the relevant mapping table probe: just about any recipient or sender attribute likely to be of interest could be made available to the mapping probe.

For a similar feature of adding LDAP attribute values to [FROM_ACCESS](#) or [recipient address-based access mapping tables](#), see the `include_spares1` MTA option.

39.27.2.4 Internal size MTA options: `map_names_size` (1-1000000)

The `map_names_size` option specifies the size of the mapping table name table, and thus the total number of [mapping tables](#) cannot exceed this number. The default is 32; the maximum allowed with normal, automatic resizing is 5000 (controlled from the `maximum.dat` file); the "hard" maximum is 1,000,000. Attempts to set this option higher than its maximum of 1,000,000 will result in an `mm_init` error, "MAP_NAMES_SIZE exceeds maximum".

Attempts to specify more mapping tables than allowed by this option's maximum (normally the automatic resizing maximum, unless a higher maximum has been explicitly configured) will result in an `mm_init` error, "no room in table for mapping named ...". In particular, note the following. In its literal meaning, the "no room in table for mapping named ..." error indicates that your configuration's current MTA internal table sizes are not large enough for your current number of mapping tables. However, also note that formatting errors in the MTA mapping file can cause the MTA to think that you have more mapping tables than you really have; for instance, check that mapping table entries are all properly indented.

39.27.2.5 Miscellaneous mapping table MTA options: `message_save_copy_flags` (bitmask)

(New in JES MS 6.3-0.01) The `message_save_copy_flags` MTA option controls whether certain additional, optional fields are included in [MESSAGE-SAVE-COPY mapping table](#) probes. The option takes a bit-encoded integer value. The following bits are defined:

Table 39.28 `message_save_copy_flags` option bit values

Bit	Value	Usage
0	1	Include transport and application information fields as the first and second, respectively, fields of the mapping table probe
1	2	Include the source channel as an additional field in the mapping table probe, (between the application information field and conversion tag field, if those fields are enabled)
2	4	Include conversion tag(s) as an additional field in the mapping table probe, immediately after the source channel field
3	7	Include the MT-PRIORITY value for this message (an integer between -9 and 9) in the mapping table probe, immediately after the conversion tag field

Bit 0 is the least significant bit.

The default is 0.

39.27.2.6 Miscellaneous mapping table MTA options: `original_channel_probe` (0 or 1)

RESTRICTED.

The `original_channel_probe` MTA option controls whether things like [CONVERSIONS mapping table](#) probes, when performed by the [conversion channel](#) and any additional `conversion_*` channels, or the [process channel](#) or [hold channel](#), use the original channel as the input channel name, or use the current source channel as the input channel name. For instance, this distinction can make a difference when applying both conversion and character set conversions to the same message. Conversions happen before character set conversion. So this option affects which channel name should appear in the `IN-CHAN=channel-name` clause of the [CHARSET-CONVERSION mapping](#) probe, the original channel, or the current source channel (which will be the conversion channel at that point). The default is 0, meaning to use the current input channel name.

39.27.2.7 Miscellaneous mapping table MTA options: `use_comment_strings` (bitmask)

New in JES MS 6.1. The `use_comment_strings` MTA option takes a bit encoded integer argument. If bit 0 (value 1) of this option is set, then the familiar

`source-channel|destination-channel|`

prefix will be included in [COMMENT_STRINGS](#) mapping table probes. See the [commentmap](#) and [sourcecommentmap](#) channel options, and [RFC 822 comments strings and personal name modification](#) for more discussion of the `COMMENT_STRINGS` mapping table.

39.27.2.8 Alias and address MTA options: `use_forward_database` (bitmask)

The `use_forward_database` MTA option controls whether or not the MTA makes use of the forward database, and also controls the exact format of probes of the forward database and [FORWARD mapping table](#). The value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 39.29 `use_forward_database` MTA option bits

Bit	Value	Usage
0	1	When set, the forward database is used.
3	8	When set, channel-level granularity is used with the forward database entries. Forward database entries' left hand sides must have the form (note the vertical bars,) <i>source-channel from-address to-address</i> Note that source-specific forwarding is very seldom appropriate, and in those rare cases where it is appropriate, the forward database is seldom the most suitable choice for achieving it.
4	16	When set, channel-level granularity is used with the FORWARD or any domain catchall mapping . The mapping entries' patterns (left hand sides) must have the form (note the vertical bars,) <i>source-channel from-address to-address</i> Note that source-specific forwarding is very seldom appropriate, and in those rare cases where it is appropriate, the FORWARD mapping is seldom the most suitable choice for achieving it.
5	32	When set, modifies the effect of bit 3 (source-specific forward database probes) by also including the destination channel in the probe.
6	64	When set, modifies the effect of bit 4 (source-specific FORWARD or domain catchall mapping probes) by also including the destination channel in the probe.
7	128	(New in 8.0) When set, includes the initial address presented for alias processing in the FORWARD mapping probe. This address appears immediately before the intermediate address included by bit 8 below.
8	256	(New in 8.0) When set, include the current intermediate address in the FORWARD mapping probe. This address appears immediately before the final recipient address.
9	512	(New in 8.0) When set, include the authenticated sender address address in the FORWARD or any domain catchall mapping probe. This address appears immediately after the destination channel and before conversion tags.

Bit 0 is the least significant bit.

The default value for `use_forward_database` is 0, which means that the MTA will not use the forward database at all. Note that a [FORWARD](#) mapping table, if present, is always consulted.

39.27.2.9 Alias and address MTA options: use_reverse_database (bitmask)

The use_reverse_database MTA option controls whether or not the MTA makes use of the [address reversal database](#) and [REVERSE mapping table](#) as a source of substitution addresses. (Note that it can not disable use of any [reverse_url](#) setting, although its bit 2, value 4, does affect the scope of reverse_url application.) Its value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 39.30 use_reverse_database MTA option bits

Bit	Value	Usage
0	1	When set, reverse database based address reversal is applied to addresses after they have been rewritten by the MTA address rewriting process; (this does not affect reverse_url or REVERSE mapping table based address reversal subsequent to address rewriting, which is controlled merely by the existence of a reverse_url setting or existence of a REVERSE mapping table, respectively).+
1	2	When set, reverse database and/or REVERSE mapping based address reversal is applied before addresses have had MTA address rewriting applied to them; (this does not affect reverse_url based address reversal, which always occurs after rewriting has been applied; for the REVERSE mapping, setting this bit causes an additional consultation of the REVERSE mapping prior to address rewriting, in addition to the subsequent to rewriting consultation which will always be performed).+
2	4	When set, address reversal, including the reverse_url option setting if applicable, will be applied to all (except envelope To) addresses, including forward-pointing header addresses, not just to backward-pointing addresses.
3	8	When set, channel-level granularity is used with the REVERSE mapping . REVERSE mapping table (pattern) entries must have the form (note the vertical bars,) <i>source-channel destination-channel address</i>
4	16	When set, channel-level granularity is used with address reversal database entries. Reversal database entries' left hand sides must have the form (note the vertical bars,) <i>source-channel destination-channel address</i>
5	32	Apply REVERSE mapping even if a reverse database entry has already matched.
6	64	Apply address reversal to message ids; see Internal host names in Received: and Message-Id: header lines for an example.
7	128	When set, this modifies the effect of bit 4 (channel-level granularity of address reversal database entries); when this bit is also set, the address reversal database entries take the form (note the vertical bars,) <i>destination-channel address</i>
8	256	When set, this modifies the effect of bit 3 (channel-level granularity of REVERSE mapping table entries); when this bit is also set, the REVERSE mapping table entries take the form (note the vertical bars,) <i>destination-channel address</i>

10	1024	During subsequent-to-rewriting address reversal, (that is, that reversal due to having bit 0 (value 1) set), this disables the normal initial reverse database lookup, though a database lookup can still be caused by a \$D in the REVERSE mapping table
11	2048	During prior-to-rewriting address reversal, (that is, that reversal due to having bit 1 (value 2) set), this disables the normal initial reverse database lookup, though a database lookup can still be caused by a \$D in the REVERSE mapping table
12	4096	(New in 8.0.) When set, include the name of the header field the address being processed came from in the mapping probe, delimited by vertical bars, immediately after source channel, destination channel, and conversion tag information. A trailing colon is always included in the field name. A blank name appears when envelope addresses are being processed.
13	8192	(New in 8.0.) When set, do not impose source block and recipient limits and capture actions based on the envelope from (MAIL FROM) address. For background discussion, see Intended side effects of LDAP address reversal .
14	16384	(New in 8.0.) When set, do not impose source block and recipient limits and capture actions based on the authenticated sender address. For background discussion, see Intended side effects of LDAP address reversal .

+In initial iMS 5.2 and earlier versions, the 0th and 1st bits of `use_reverse_database` not only controlled when, but also *whether* a [REVERSE mapping table](#) would be consulted at all for address rewriting; now if a [REVERSE mapping table](#) exists, it definitely *will* be consulted for address reversal (at least) subsequent to address rewriting; (depending upon bit 1, it may also be consulted prior to address rewriting). So this is a change from iMS 5.2 and earlier versions.

Bit 0 is the least significant bit.

The default value for `use_reverse_database` is 5, which means that in addition to consulting any [reverse_url](#) setting to reverse envelope From addresses and both backwards and forwards pointing header addresses after they have passed through the normal address rewriting process, the MTA will also consult any [reverse database](#) or [REVERSE mapping](#) to reverse Envelope From addresses and both backwards and forwards pointing header addresses after they have passed through the normal address rewriting process. Simple address strings are presented to both the [REVERSE mapping](#) and the reverse database. Note that a value of 0 disables the use of the address reversal completely. (Note that the default of 5 represents a change from early versions of PMDF in which this option had a default value of 1 (reverse only backwards pointing addresses).)

39.27.2.10 Miscellaneous mapping table MTA options: `use_personal_names` (bitmask)

New in JES MS 6.1. The `use_personal_names` MTA option takes a bit encoded integer argument. If bit 0 (value 1) of this option is set, then the familiar

`source-channel|destination-channel|`

prefix will be included in [PERSONAL_NAMES](#) mapping table probes. See the [personalmap](#) and [sourcepersonalmap](#) channel options, and [RFC 822 comments strings and personal name modification](#) for more discussion of the [PERSONAL_NAMES](#) mapping table.

Setting bit 1 (value 2) causes the inclusion of the name of the header field the address being processed came from in the mapping probe, immediately after source channel, destination

channel, and conversion flag information. A trailing colon is always included in the field name.

If bit 2 (value 4) is set any personal name generated during address reverse will not be used by default. Instead the revised name will be added to the `PERSONAL_NAMES` mapping probe immediately following the original personal name associated with the address. If the mapping wishes to use the revised name all it needs to do is return that name and set `$Y`.

39.28 Memcache MTA options

New in the 8.0 release, the MTA supports use of memcache for certain database/storage uses. `memcache_*` MTA options control the MTA's connections to memcache. The `*_database_url` MTA options, when set to [memcache: URL](#) values, configure use of Memcache to store MTA databases. See also the [enable_sieve_memcache](#) MTA option, which enables Sieve filter use of a [memcache](#) operator in Sieve scripts.

39.28.1 Memcache MTA options: `memcache_host` (host)

The `memcache_host` MTA option specifies the host name (or IP address) of the host providing memcached service. There is no default.

39.28.2 Memcache MTA options: `memcache_port` (port)

The `memcache_port` MTA option specifies the port number for the memcached server. If not specified, it defaults to 11211, the usual port for memcache servers.

39.28.3 Memcache MTA options: `memcache_expire` (integer)

The `memcache_expire` MTA option specifies the amount of time, in seconds, that a connection to a memcached server can remain idle and still be eligible for reuse.

39.28.4 Memcache MTA options: `memcache_timeout` (integer)

The `memcache_timeout` MTA option specifies the amount of time, in seconds, to wait for a connection to the memcached server.

39.28.5 Memcache MTA options: `alias_database_url` (memcache: URL)

The value of the `alias_database_url` MTA option should be a [memcache: URL](#) for storing [alias database](#) data.

39.28.6 Memcache MTA options: `domain_database_url` (memcache: URL)

The value of the `domain_database_url` MTA option should be a [memcache: URL](#) for storing [domain database](#) data (that is, supplementary [rewrite rules](#)).

39.28.7 Memcache MTA options: `forward_database_url` (memcache: URL)

The value of the `forward_database_url` MTA option should be a [memcache: URL](#) for storing [forward database](#) data. Note that `forward_database_url` will only be consulted ([memcache](#) will only be consulted) if bit 2 (value 4) of the `use_text_databases` MTA option is clear (*not* set); setting `use_text_databases` to a value subsuming 4 causes any setting of `forward_database_url` to be ignored.

39.28.8 Memcache MTA options: `general_database_url` (memcache: URL)

The value of the `general_database_url` MTA option should be a [memcache: URL](#) for storing [general database](#) data. Note that `general_database_url` will only be consulted ([memcache](#) will only be consulted) if bit 0 (value 1) of the `use_text_databases` MTA option is clear (*not* set); setting `use_text_databases` to a value subsuming 1 causes any setting of `general_database_url` to be ignored.

39.28.9 Memcache MTA options: `reverse_database_url` (memcache: URL)

The value of the `reverse_database_url` MTA option should be a [memcache: URL](#) for storing [reverse database](#) data. Note that `reverse_database_url` will only be consulted ([memcache](#) will only be consulted) if bit 1 (value 2) of the `use_text_databases` MTA option is clear (*not* set); setting `use_text_databases` to a value subsuming 2 causes any setting of `reverse_database_url` to be ignored.

39.28.10 Memcache MTA options: `ssr_database_url` (memcache: URL)

The value of the `ssr_database_url` MTA option should be a [memcache: URL](#) for storing Server Side Rules data, (that is, user Sieve filters stored in an MTA "database").

Note that the entire facility for storing Sieve filters in a Server Side Rules database is mostly obsolete nowadays, as nowadays user Sieve filters are normally stored in an [LDAP attribute](#) within users' own LDAP entries.

39.29 Message archival and hashing MTA options

The MTA options discussed in this section relate to the envelope "journal" format captured messages that the MTA can optionally generate, as well as to the message hashes optionally generated by the MTA in order to facilitate integration with alternative message archiving

software -- with regards to which, see also the [Message Store archive options](#) and [Archiving messages](#). The `journal_format` MTA option would be used when "journal" style message capture is in use (due to Sieve "capture :journal" actions, or due to LDAP attribute based message capture with optional "journal" format configured). The message hashing options would be used in conjunction with the [message hash channel options](#).

39.29.1 Archive message format control: `journal_format` (bitmask)

The `journal_format` MTA option controls the [format](#) of Microsoft® Exchange journaling messages generated by the MTA. This is a bit-encoded option. Currently assigned bits are:

Table 39.31 `journal_format` MTA option bit values

Bit	Value	Description
0	1	If set, generate the basic 2007 journal format instead of the 2003 format.
1	2	If set, set the From:/To:/Subject: of the journal message to be the same as the message being journaled. Note that setting this may cause looping problems for setups that use header checks to determine what messages to archive.
2	4	If set, generate a X-MS-Exchange-Organization-Journal-Report: header field rather than a X-MS-Journal-Report: field.
3	8	If set, include expanded/forwarded address information in the report (if such information is available -- see the addrtypescan channel option). Note that bit 0 must also be set for this to work.

The default value for this option is 0. This option is intended to facilitate interoperating with Microsoft Exchange itself, in particular, so that MTA-generated journal messages can be imported into Microsoft Exchange.

39.29.2 Message archiving and hashing options: `message_hash_algorithm` (hash algorithm name)

The `message_hash_algorithm` MTA option controls what hash algorithm the MTA uses to generate a hash over the message. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, HAVAL, MD128, MD160, or TIGER. The default is MD5.

39.29.3 Message archiving and hashing MTA options: `message_hash_fields` (list of header names)

This option takes either a comma-separated list or space-separated list of up to thirty-two known (to the MTA) header field names, each with or without a trailing colon (and intervening spaces are okay too). The default is:

```
Message-id: ,From: ,To: ,Cc: ,Bcc: ,Resent-message-id: ,Resent-From: ,Resent-To: ,  
Resent-Cc: ,Resent-Bcc: ,Subject: ,Content-id: ,Content-type: ,Content-Description:
```

(The Content-type: and Content-description: here are those from the top or outer MIME level of the message.)

See [Message identifier generation](#) for a discussion of use of this option in the content of message archiving.

39.29.4 Message archiving and hashing MTA options: unique_id_template (string)

The `unique_id_template` MTA option specifies a template used to convert an address for a "local" user into a [unique identifier](#). The template's substitution vocabulary is the same as that for delivery options; (see the [delivery_options](#) MTA option, and especially the permitted [LDAP URL substitution sequences](#)). The resulting unique identifier is intended for use by message archiving tools. So for instance a site could set

```
unique_id_template=$M@$D
```

to use each user's `uid@domain` as that user's unique identifier.

39.30 Message size MTA options

The MTA has a number of options relating to message size, such as limits on the size of messages allowed in by the MTA, message size affecting message processing priority, limits on the extent to which the MTA looks into (processes) messages of complex MIME structure, and fine tuning of message fragmentation. There are also MTA options relating specifically to the size of notification messages. And there are MTA options controlling the error text returned when messages are "too large" in one sense or another.

See also the [maxprocchars](#) channel option.

39.30.1 Message size MTA options: block_limit (integer)

The `block_limit` MTA option places an absolute limit on the size, in [MTA blocks](#), of any message which may be sent by or received with the MTA. Any message exceeding this size will be rejected. By default, the MTA imposes essentially no size limits; (more precisely, the default is the maximum allowed integer, $2^{31}-1$). Note also that the `blocklimit` and `sourceblocklimit` channel options can be used to impose limits on a per-channel basis. The size in bytes of an MTA block is specified with the `block_size` MTA option.

The `line_limit` MTA option allows for a similar limit, expressed in terms of lines in a message.

Note that domain LDAP attributes can be used to impose per-domain limits; see the [ldap_domain_attr_blocklimit](#) MTA option (normally specifying the use of the `mailDomainMsgMaxBlocks` LDAP attribute) and

[ldap_domain_attr_sourceblocklimit](#) MTA option. Or user LDAP attributes can be used to impose per-user limits; see the [ldap_maximum_message_size](#) MTA option (normally specifying the use of the `mgrpMsgMaxSize` LDAP attribute) and the [ldap_sourceblocklimit](#) MTA option.

39.30.2 Message size MTA options: `block_size` (integer > 0)

The MTA measures message size in units of MTA "blocks". For instance, the [MTA message transaction log file](#) (resulting from placing the [logging](#) option on a channel) records message sizes in terms of blocks. MTA blocks are also the unit of measurement for various message size limit and message size based effects, as specified via channel options including [blocklimit](#), [sourceblocklimit](#), [alternateblocklimit](#), [holdlimit](#), [nonurgentblocklimit](#), [normalblocklimit](#), [urgentblocklimit](#), and [maxblocks](#), and via MTA options including [block_limit](#), [bounce_block_limit](#), [content_return_block_limit](#), [header_limit](#), [log_size_bins](#), [non_urgent_block_limit](#), [normal_block_limit](#), [urgent_block_limit](#), [max_header_block_use](#), and [max_internal_blocks](#), and via LDAP attributes named by MTA options including [mailMsgMaxBlocks](#) ([ldap_blocklimit](#)), [ldap_sourceblocklimit](#), [mgrpMsgMaxSize](#) ([ldap_maximum_message_size](#)), [mailDomainMsgMaxBlocks](#) ([ldap_domain_attr_blocklimit](#)), and [ldap_domain_attr_sourceblocklimit](#), and the alias option [alias_blocklimit](#). Normally an MTA block is equivalent to 1024 octets. This option can be used to modify this sense of what a block is.

NOTE: The MTA stores message sizes internally as an integer number of blocks. If the size of a block in bytes is set to a very small value it is possible for a very large message to cause an integer overflow. A message size of greater than 2^{31} blocks would be needed, but this value is not inconceivable if the block size is small enough.

Given the extensive list (above) of values measured in units of "blocks", it may be useful here to also list values that are *not* measured in "blocks". In particular, measurements that do *not* use the `block_size`, but which instead are always measured in units of bytes, include the [conversion channel](#) environment variable `PART_SIZE`, the Content-length: MIME header line value, the SMTP protocol extension `SIZE` value (see [RFC 1870](#)), and the user-level [mailQuota](#) LDAP attribute (more properly from the MTA's point of view, the attribute pointed to by the [ldap_disk_quota](#) MTA option) and the domain-level `mailDomainDiskQuota` LDAP attribute (which note is not used by the MTA proper). Furthermore, the user-level [mailQuota](#) LDAP attribute (more properly from the MTA's point of view, the attribute pointed to by the [ldap_disk_quota](#) MTA option) and the Sieve filter `size test`'s value are also normally interpreted as bytes, (though Sieve `size` values can optionally use a K, M, or G unit indicator to indicator measuring in units of 2^{10} , 2^{20} , or 2^{30} , and similarly the [mailQuota](#) attribute's value can use K, M, or G unit indicators).

39.30.3 Notification message and return job options (`bounce_block_limit`)

The `bounce_block_limit` MTA option may be used to force bounces of messages over the specified size (number of blocks, as defined via the [block_size MTA option](#)) to return only the message headers, rather than the full message content. This overrides any NOTARY ([RFC 1891](#)) setting originally present on original messages.

By default, there is essentially no limit; (more precisely, the default is the maximum allowed integer, $2^{31}-1$).

39.30.4 Notification message MTA options: `content_return_block_limit` (non-negative integer)

The `content_return_block_limit` MTA option may be used to force on the NOTARY (RFC 1891) non-return of content flag for messages over the specified size (in units of MTA blocks); if such a message is subsequently bounced by a system that supports NOTARY, then the original message contents will not be included in the bounce message. By default, this option is not set and hence entire original messages of arbitrary size potentially may be included in bounce messages; however, even when this option is not set, the MTA will automatically truncate original message content when generating a notification message if a message size limit (e.g., the `blocklimit` channel option) has been imposed on the relevant destination channel.

This option only applies to messages that do not have their own explicit NOTARY (RFC 1891) setting controlling return of content; when NOTARY has been used, it takes precedence over this option.

39.30.5 Message size MTA options: `header_limit` (integer)

The `header_limit` option sets a limit, in MTA blocks (see the `block_size` MTA option), on how big a message's primary (outermost) header can be. Headers over the specified size will be (silently) truncated. The default is 2000 blocks.

Source channels may have their own, more restrictive, `headerlimit` set; each channel will minimize its own setting with this general, MTA-wide `header_limit` setting.

Compare with the `max_header_blocks` and `max_header_lines` MTA options, which instead of silent truncation in the form of deletion of the header lines, cause silent "truncation" by forcing the excess header material into the message body. And see also the `maxprocchars` channel option, limiting how much of the header the MTA will process. And for yet more approaches to header limits and header truncation, see the `maxheaderadds` and `maxheaderchars` channel options, and the `MAXCHARS`, `MAXIMUM`, and `MAXLINESheader trimming options`.

39.30.6 Message size MTA options: `line_limit` (integer)

The `line_limit` MTA option places an absolute limit on the overall number of lines in any message which may be sent or received by the MTA. Any message exceeding this limit will be rejected. By default, the MTA imposes essentially no line count limits; (more precisely, the default limit is the maximum allowed integer, $2^{31}-1$). Note also that the `linelimit` channel option can be used to impose line limits on a per-channel basis. Similarly, the `block_limit` MTA option may be used to set a limit based on number of blocks in a message.

39.30.7 Message size MTA options: `local_quota_checks` (integer)

`max_header_block_use`,
`max_header_line_use` MTA
options

RESTRICTED: The `local_quota_checks` MTA option is not fully implemented.

39.30.8 Message fragmentation size limit MTA options: `max_header_block_use` (real number strictly between 0 and 1), `max_header_line_use` (real number strictly between 0 and 1)

39.30.8.1 `max_header_block_use`

The `max_header_block_use` MTA option controls what fraction of the available [message blocks](#) can be used by message headers. The default is 0.5.

This MTA option is used to fine-tune the message fragmentation effect of the [maxblocks](#) channel option.

39.30.8.2 `max_header_line_use`

The `max_header_line_use` MTA option controls what fraction of the available message lines can be used by message headers. The default is 0.5.

This MTA option is used to fine-tune the message fragmentation effect of the [maxlines](#) channel option.

39.30.9 Message size MTA options: `max_header_blocks` (integer)

The `max_header_blocks` MTA option causes truncation of the original, incoming message header after the specified number of blocks (by forcing additional, supposedly header material, into the message body). The default is no limit, but see also the [max_header_lines](#) and [header_limit](#) MTA options. The `max_header_blocks`, `max_header_lines`, and `header_limit` MTA options provide some protection against denial-of-service attacks in the form of messages with extravagantly large/long headers.

See also the [maxprocchars](#) channel option, limiting how much of the header the MTA will process.

39.30.10 Message size MTA options: `max_header_lines` (integer)

The `max_header_lines` MTA option causes truncation a of message's original, incoming message header after the specified number of lines (by forcing additional, supposedly header material, into the message body). The default is 10,000. The [max_header_blocks](#), [header_limit](#), and `max_header_lines` MTA options provide some protection against denial-of-service attacks in the form of messages with extravagantly large/long headers.

See also the [maxprocchars](#) channel option, limiting how much of the header the MTA will process.

39.30.11 Maximum message levels/parts to process MTA options: `max_mime_levels` (integer), `max_mime_parts` (integer)

39.30.11.1 `max_mime_levels`

Specify the maximum depth to which the MTA should process MIME messages. The default is 100, meaning that the MTA will process up to one hundred levels of message nesting. Higher values may require additional amounts of memory and, [for the Dispatcher, additional per-thread storage space](#). If `max_mime_levels`>0, then levels of nesting higher than specified will not be processed. As of 8.0, if `max_mime_levels`<0, then in addition such messages will be sidelined as `.HELD` files. See the [held_sndopr](#) MTA option for discussion of optionally logging (to syslog) when such events occur.

39.30.11.2 `max_mime_parts`

Specify the maximum number of MIME parts which the MTA should process in a MIME message. The default value is the maximum allowed integer, 2147483647: essentially no limit is imposed. If `max_mime_parts`>0, then message parts greater than specified will not be processed. As of 8.0, if `max_mime_parts`<0, then in addition such messages will be sidelined as `.HELD` files. See the [held_sndopr](#) MTA option for discussion of optionally logging (to syslog) when such events occur.

39.30.12 Size effects of message priority MTA options: `non_urgent_block_limit` (integer), `normal_block_limit` (integer), `second_class_block_limit` (integer), `urgent_block_limit` (integer)

Note that as of the 8.0 release, these size-based priority override MTA option effects are nullified if the MT-PRIORITY SMTP extension has been used to set an explicit priority value.

39.30.12.1 `non_urgent_block_limit`

The `non_urgent_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to lower than non-urgent priority, meaning that they will not be processed immediately and will wait for processing until the next periodic delivery run. The value is interpreted in terms of MTA blocks, as specified by the [block_size](#) MTA option. Note also that the [nonurgentblocklimit](#) channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.30.12.2 `normal_block_limit`

The `normal_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to non-urgent priority. The [Job Controller automatically pays attention to message effective processing priority](#) when scheduling delivery attempts. The value is interpreted in

terms of MTA blocks, as specified by the `block_size` MTA option. Note also that the `normalblocklimit` channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.30.12.3 `second_class_block_limit`

The `second_class_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to third class priority. The value is interpreted in terms of MTA blocks, as specified by the `block_size` MTA option. Note also that the `secondclassblocklimit` channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.30.12.4 `urgent_block_limit`

The `urgent_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to normal priority. The `Job Controller automatically pays attention to message effective processing priority` when scheduling delivery attempts. The value is interpreted in terms of MTA blocks, as specified by the `block_size` MTA option. Note also that the `urgentblocklimit` channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.31 Message tracking MTA options

New in 8.0 is support for Message Tracking. There are several MTA options relating to Message Tracking.

The `log_tracking` MTA option enables inclusion of tracking/recall information in MTA message transaction log entries.

The `tracking_debug` MTA option enables low-level debugging (typically only meaningful to Oracle support) regarding the MTA's message tracking operation.

The `ldap_auth_attr_recall_secret` MTA option specifies the name of the LDAP attributes where a user's general recall secret is stored.

39.31.1 Tracking Information Storage (`tracking_mode`)

The `tracking_mode` MTA option controls how `message tracking` information is stored by the MTA. The default value of 0 disables storage of tracking information, while setting `tracking_mode` to 1 enables storage using a `memcached server` shared across the deployment. The use of other values is currently restricted.

39.31.2 Tracking Update Retry Control (`tracking_retries`, `tracking_retry_delay`)

Tracking information updates can fail because of simultaneous access attempts to the underlying database. If this happens the update can be retried. The `tracking_retries` MTA option specifies how many times to retry the update. The default is 5 retries.

The `tracking_retry_delay` MTA option specifies the amount of time to delay between retry attempts in units of centiseconds. The default is 10 centiseconds.

39.32 MeterMaid MTA options

MTA options for setting general [MeterMaid](#) configuration parameters were introduced in Messaging Server 7.2-7.02; previously (from JES MS 6.3-0.15 onwards), only `configutil` parameters had been available for such settings.

The general MeterMaid configuration `metermaid_*` MTA options exist to control fundamentals of MeterMaid operation such as the host and port on which MeterMaid is running, *etc.* See also the `enable_sieve_metermaid` MTA option which permits [Sieve filters](#) to use a "metermaid" operator directly.

39.32.1 MeterMaid MTA options: `metermaid_backoff` (integer)

The `metermaid_backoff` MTA option specifies the frequency, in seconds, with which the MTA connects to [MeterMaid](#); for MTA purposes, this MTA option if set will override the legacy configuration `metermaid.mtaclient.connectfrequency` `configutil` parameter, or its Unified Configuration equivalent, the `connectfrequency` MeterMaid MTA client option. If neither the MTA option, nor the (legacy configuration) `configutil` parameter or (Unified Configuration) MeterMaid MTA client option is set, then the default is 15.

39.32.2 MeterMaid MTA options: `metermaid_expire` (integer)

The `metermaid_expire` MTA option specifies the idle time, in seconds, that the MTA permits for a connection to [MeterMaid](#); after this time period, the MTA will expire the connection.

39.32.3 MeterMaid MTA options: `metermaid_host` (hostname)

The `metermaid_host` MTA option specifies the [MeterMaid](#) host for the [Sieve metermaid operator](#). This MTA option if set will override the legacy configuration `metermaid.config.serverhost` `configutil` parameter, or its Unified Configuration equivalent, the `server_host` MeterMaid client option. There is no default.

39.32.4 MeterMaid MTA options: `metermaid_port` (port)

The `metermaid_port` MTA option specifies the [MeterMaid](#) port for the [Sieve metermaid operator](#). This MTA option if set overrides the legacy configuration `metermaid.config.port` `configutil` parameter, or its Unified Configuration equivalent, the `port` MeterMaid option. If neither the MeterMaid option nor `configutil` parameter/MeterMaid option is set, then the default is 63837.

39.32.5 MeterMaid MTA options: `metermaid_secret` (string)

The `metermaid_secret` MTA option specifies the secret string or strings used to verify [MeterMaid](#) communications; for the Sieve `metermaid` operator, this MTA option if set

metermaid_timeout MTA
option

overrides the legacy configuration `metermaid.config.secret` configutil parameter, or its Unified Configuration equivalent, the [secret](#) MeterMaid option. There is no default.

This option can either contain a single secret or a series of host-secret pairs. In the latter case, the general format is:

```
host1:secret1,host2:secret2,...,hostN:secretN
```

The secret to use is selected by comparing the current Metermaid host with each host pattern on the list until a match is found. Glob-style wildcards can be used in the host patterns.

39.32.6 MeterMaid MTA options: `metermaid_timeout` (integer)

The `metermaid_timeout` MTA option specifies the timeout, in seconds, for receiving data from [MeterMaid](#); for MTA purposes, this MTA option if set overrides the legacy configuration `metermaid.mtaclient.readwait` configutil parameter, or its Unified Configuration equivalent, the [timeout](#) MeterMaid MTA client option. If neither the MTA option nor the configutil parameter/MeterMaid MTA client option is set, then the default is 10.

39.33 MLS MTA options

RESTRICTED. Not yet fully implemented.

A couple of MTA options affect MLS (Multi Level Security) processing by the MTA. The main one is [mls](#); see also the [ldap_mlrange](#) MTA option which specifies the name of the user-level LDAP attribute that stores a user's MLS range, and the [error_text_mls_access_failure](#) MTA option, which controls the exact error text returned when an MLS access failure occurs.

See also the [mlslabel](#) and [mlsrange](#) channel options.

39.33.1 `mls` Option

RESTRICTED. Not yet fully implemented.

39.34 MTQP MTA options

New in the 8.0 release is MTQP (Message Tracking and Query Protocol) support.

39.34.1 MTQP MTA options: `mtqp_port` (port)

New in MS 8.0.

39.34.2 MTQP MTA options: `mtqp_timeout` (integer)

New in MS 8.0.

39.34.3 MTQP MTA options: `mtqp_expire` (integer)

New in MS 8.0.

39.35 Notification message MTA options

The MTA has a number of options relating to notification messages: the timing of their generation, the size of notification messages, the content of notification messages, the postmaster address visible in notification messages, *etc.*

For hosted-domain-specific postmaster addresses, see the `ldap_domain_attr_report_address` MTA option.

For the channel-specific timing of generation of notification messages, see especially the `notices` channel option, as well as the `backoff` channel option.

For detailed discussion of the format of notification messages, see the general discussion under [Notification messages](#).

The `return_split_period` and `return_cleanup_period` MTA options control the frequency at which the MTA performs certain tasks related to the MTA return job. The `return_debug` and `return_verify` MTA options enable, respectively, low-level debugging and shell script logging regarding the MTA's return job.

39.35.1 Notification message and return job options (`bounce_block_limit`)

The `bounce_block_limit` MTA option may be used to force bounces of messages over the specified size (number of blocks, as defined via the [block_size MTA option](#)) to return only the message headers, rather than the full message content. This overrides any NOTARY ([RFC 1891](#)) setting originally present on original messages.

By default, there is essentially no limit; (more precisely, the default is the maximum allowed integer, $2^{31}-1$).

39.35.2 Notification message MTA options: `content_return_block_limit` (non-negative integer)

The `content_return_block_limit` MTA option may be used to force on the NOTARY ([RFC 1891](#)) non-return of content flag for messages over the specified size (in units of MTA blocks); if such a message is subsequently bounced by a system that supports NOTARY, then the original message contents will not be included in the bounce message. By default, this option is not set and hence entire original messages of arbitrary size potentially may be included in bounce messages; however, even when this option is not set, the MTA will automatically truncate original message content when generating a notification message if a message size limit (*e.g.*, the `blocklimit` channel option) has been imposed on the relevant destination channel.

This option only applies to messages that do not have their own explicit NOTARY ([RFC 1891](#)) setting controlling return of content; when NOTARY has been used, it takes precedence over this option.

39.35.3 Notification message MTA options: `history_to_return` (1-200)

The `history_to_return` MTA option controls exactly how many delivery attempt history records are included in returned messages, when `return_delivery_history` is set to enable such inclusion. The delivery history provides some indication of how many delivery attempts were made and in some cases indicates the reason the delivery attempts failed. The default value for this option is 20.

39.35.4 Notification message MTA options: `lines_to_return` (integer)

The `lines_to_return` MTA option controls how many lines of message content the MTA includes when generating a notification message for which it is appropriate to return only a sample of the contents. The default is 20. Note that this option is irrelevant when generating a NOTARY bounce message, where either the full content or merely headers are included, according to the choice specified during the initial submission of the message. So in practice, this option is mostly only relevant to the warning messages the MTA's return job sends about messages awaiting further delivery retries in the MTA's message queue area.

Note that setting `lines_to_return=0` will cause the warning messages generated by the MTA regarding not-yet-delivered messages to contain only message headers (none of the body of the original message).

39.35.5 Notification message MTA options: `notary_decode` (-1, 0, or 1)

When the MTA is generating a DSN and using a `%H` substitution sequence to insert original message headers into the DSN, this option controls whether encoded-words (*i.e.*, material in character sets other than US-ASCII) present in the original message header being inserted due to `%H` are left alone, decoded if already in the character set specified in the `return_prefix.txt` file, or forcibly converted to the `return_prefix.txt` character set and then decoded. (Note that the character set specified in the `return_prefix.txt` file is the character set used for the first, human readable portion of the DSN.)

The default value is 0, meaning that encoded-words that match the character set specified in the `return_prefix.txt` file will be decoded; encoded-words in other character sets will be left in literal encoded form. A value of 1 causes encoded-words present in the original message header to be first converted to the character set specified in the `return_prefix.txt` file and then decoded; that is, assuming that the character sets are all compatible, all material will be converted and decoded and no encoded-words will be left. A value of -1 disables decoding (and conversion) unconditionally; the original message headers are substituted literally, including the encoded-words in their literal (MIME encoded) form.

Caution should be exercised in setting this option to a value of 1, as information loss can occur, with resultant confusion, when a rich character set such as UTF-8 is converted to a more limited character set (a character set lacking many characters present in the original character set) such as ISO-8859-1 or US-ASCII.

39.35.6 Notification message MTA options: `notary_quote` (1-127)

The `notary_quote` MTA option specifies the ASCII representation of the character that marks substitution sequences in `return_*.txt` files and `disposition_*.txt` files. It

defaults to 25 (the ASCII position of the percent character) so substitutions are %R, %u, *etc.*, as listed in [return_*.txt file substitution sequences](#).

39.35.7 Notification message MTA options: **return_address (address)**

The `return_address` option sets the return address for the local Postmaster. By default, the local Postmaster's address is `postmaster@localhost`, where the `localhost` corresponds to the `msconfig` setting of `channel:1.official_host_name` (which itself typically references the `instance.base.hostname` setting via a macro substitution). The `return_address` MTA option provides a way to override this default with the address of your choice.

Care should be taken in the selection of this postmaster address, as an illegal selection may cause rapid message looping and pile-ups of huge numbers of spurious error messages. See the [returnaddress](#) channel option for discussion of overriding this address on a per-channel level, or the [ldap_domain_attr_report_address](#) MTA option specifying a domain-level LDAP attribute which can be used to override this address on a per-domain basis. See also the [return_personal](#) MTA option, which sets the Postmaster's personal name (as opposed to their actual mailbox address).

39.35.8 Notification message MTA options: **return_delivery_history (0 or 1)**

The `return_delivery_history` MTA option controls whether or not a history of delivery attempts is included in returned messages. The delivery history provides some indication of how many delivery attempts were made and in some cases indicates the reason the delivery attempts failed. A value of 1 enables the inclusion of this information and is the default. A value of 0 disables return of delivery history information. The [history_to_return](#) MTA option controls how much history information is actually returned.

39.35.9 Notification message MTA options: **return_envelope (bitmask)**

The `return_envelope` MTA option takes a bitmask value.

Bit 0 (value = 1) controls whether or not [return notifications generated by the MTA](#) are written with a blank envelope address *vs.* with the [address of the local postmaster](#). Setting the bit forces the use of the local postmaster address, while clearing the bit forces the use of a blank address. Note that the use of a blank address is mandated by [RFC 1123](#). However, some systems do not handle blank envelope From addresses properly 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. Again, this is used to accommodate non-compliant systems that don't conform to [RFC 821](#), [RFC 822](#), or [RFC 1123](#).

Bit 2 (value = 4) controls whether or not the MTA checks that any (non-empty) envelope From address matches (rewrites to) an MTA channel.

Setting bit 3 (value = 8) is the global (for all channels) equivalent of setting the [mailfromdnsverify channel option](#): it controls whether or not the MTA checks that the

domain in the envelope From address resolves in the DNS. That is, setting the bit causes the MTA to require that a DNS entry can be found corresponding to the domain in the envelope From address; but the type of DNS entry does not matter.

Setting bit 4 (value = 16) causes the MTA to enforce that if the envelope From address claims a local domain name, the envelope From address must correspond to a user address (user alias).

New in 8.0, bit 6 (value = 64) modifies the effect of setting bit 3 (value = 8) on domain validity checks. With both these bits set, if the domain in the MAIL FROM address corresponds to a null MX domain, that address will be rejected as invalid. That is, setting bit 6 causes the bit 3 domain check to also implement support for draft-delany-nullmx-01.txt.

Note also that the [returnenvelope channel option](#) can be used to impose these sorts of control on a per-channel basis.

39.35.10 Notification message MTA options: **return_personal** (RFC 2047 encoded string)

The `return_personal` option specifies the personal name to use when the MTA generates postmaster messages, *e.g.*, bounce messages. By default, the MTA uses the string "Internet Mail Delivery". The global `return_personal` value can be overridden on a per-channel basis via the [returnpersonal](#) channel option, which itself in turn can be overridden on a language-specific basis, via a language-specific [return_option.opt](#) file `RETURN_PERSONAL` option setting.

39.35.11 Notification message MTA options: **return_units** (0 or 1)

The time unit used by the message return system is controlled with the `return_units` MTA option; that is, this option controls the interpretation of the values specified for the [notices](#) channel option. A value of 0 selects units of days; a value of 1 selects units of hours. By default, units of days are used.

On UNIX and NT systems, the scheduling of the execution of the message return job is controlled by the Scheduler. In particular, with a Unified Configuration it is the `schedule.task:return_job.crontab` option that controls how frequently the return job runs: this defaults to:

```
30 0 * * * lib/return_job
```

The argument is in UNIX crontab format,

minutes hour day-of-month month-of-year day-of-week script

so the default is to run the return job every day at 30 minutes after midnight.

In legacy configuration mode (so in versions prior to 7.0.5), the scheduling of the return job was controlled via the `configutil` parameter `local.schedule.return_job`, which had a default of

```
30 0 * * * /opt/SUNWmsgsr/lib/return_job
```

with the same sort of UNIX crontab format argument.

Note: Prior to JES MS 6.0, the scheduling of the execution of the message return job was controlled by the [Job Controller](#) configuration, an approach that is now deprecated. With that approach, the scheduling was controlled by the Job Controller's `time` option in the `[periodic_job=return]` section of the `job_controller.cnf` file. For instance, to have the return job run hourly at thirty minutes past the hour, it would be set

```
[periodic_job=return]
command=IMTA_EXE:return.sh
time=00:30/1:00
```

Note that if you choose to set the [return_units](#) MTA option to a value of 1, then you will likely also need (or want) to adjust other options such as those controlling MTA transaction log file rollover.

39.35.12 Notification message MTA options: use_precedence (0 or 1)

The `use_precedence` MTA option controls whether or not the MTA makes use of the information contained in `Precedence:` header lines when deciding whether to send a delayed delivery notification message. With `use_precedence` set to 1, the default, such warning messages are not sent for messages with precedence "bulk" or "list". To instead have the MTA return job ignore the `Precedence:` header line, set `use_precedence` to 0.

39.35.13 Notification message MTA options: use_warnings_to (0 or 1)

DELETED. With the advent of standardized notification handling, this option was deprecated and is now deleted.

The `use_warnings_to` MTA option controls whether or not the MTA makes use of the information contained in `Warnings-to:` header lines when returning messages. Setting this option to 1 directs the MTA to make use of these header lines. The default is 0, which disables use of this header line. Note that this default represents a changes from the default in early versions of PMDF.

39.36 Password and TLS MTA options

There are a few options affecting the MTA's overall handling of authentication, and TLS.

39.36.1 plaintextmncipher Option Under mta

If the `plaintextmncipher` MTA option 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. This option in the `mta` group presently also applies to the MTQP and ManageSieve servers.

39.36.2 smtpproxypassword Option

The `smtpproxypassword` option specifies the password the MMP uses to authorize source channel changes on the SMTP relay servers. This option is available under `mta`, `submitproxy`, and `vdomain`. To use this functionality, that is, to use the MMP's SMTP SUBMIT Proxy, the option must be set under `mta` on the MTA back end, and must be set for the MMP's SMTP SUBMIT Proxy (thus under either `submitproxy` if being set in general, or under `vdomain` if it is only to be applied for a particular virtual domain) on a front end MMP system, and **these values must match** between front and back ends! The option has no default.

If the `mta.smtpproxypassword` option is not set, client attempts to use the XPEHLO command will receive an error (issued by the MTA SMTP server):

```
503 5.5.0 Proxy support is not enabled.
```

If `mta.smtpproxypassword` is set but its value does not match the MMP's value, client attempts to use the XPEHLO command will receive an error (issued by the MTA's SMTP server):

```
535 5.7.8 SMTP proxy authentication check failed.
```

Note that the legacy configuration equivalent of the Unified Configuration `mta.smtpproxypassword` option was the `PROXY_PASSWORD` TCP/IP-channel-specific option (which in legacy configuration, was set in the SMTP server's TCP/IP channel option file). (The MTA option `smtpproxypassword` was introduced in MS 7.0u5.) And the legacy configuration equivalent of the Unified Configuration `submitproxy.smtpproxypassword` option (as well as the `vdomain.smtpproxypassword` option) was set as `SmtProxyPassword` in the `SmtProxyAService.cfg` file.

39.36.3 sslnicknames Option Under mta

The `sslnicknames` MTA option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for MTA to offer clients if TLS is enabled. Overrides for the MTA the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter). This option in the `mta` group presently also applies to the MTQP and ManageSieve servers.

39.37 Processing priority MTA options

A few general MTA options affect message processing priority.

See also the `log_mtpriority` and `log_priority` MTA option which control, respectively, whether MTPRIORITY and effective processing priority are included in MTA message transaction log file entries. See also the `use_precedence` MTA option which controls whether Precedence: header lines affect [generation of delivery delay notifications](#). See also the `defer_group_processing` MTA option which influences whether group expansion is performed "in-line" or "off-line" (by the [Reprocess channel](#)).

For additional influences on processing priority, see also the `holdlimit` channel option, and [Job Controller priority-based processing](#), and the [Sieve setpriority and setmtpriority extensions](#).

39.37.1 Message Transfer Priority Policy: `mtpriority_policy` (string)

New in 8.0. The `mtpriority_policy` MTA option is used to specify a policy name for the handling of message transfer priorities the MTA has been configured to support. This name is announced in the SMTP EHLO response on any channel where the MT-PRIORITY extension is enabled. The default is that this option is not set, which means that no policy is announced.

Note that the MTA's Priority Assignment Policy is as follows: MT-PRIORITY values of -9,...,-4 are mapped to "non-urgent" priority; MT-PRIORITY values of -3,...,3 are mapped to "normal" priority; MT-PRIORITY values of 4,...,9 are mapped to "urgent" priority. An explicit MT-PRIORITY value specified on a submitted message will override the MTA's older priority (*e.g.*, Priority: header line based) handling, as well as any of the MTA's older size-based priority override adjustments (*e.g.*, `non_urgent_block_limit`, *etc.*). (However, a Sieve filter `setmtpriority` action can override even an explicit MT-PRIORITY value.) Messages that come in without an explicitly specified MT-PRIORITY are subject to the MTA's older priority handling, and for MT-PRIORITY purposes (such as mapping table probes including MT-PRIORITY value) will be considered to have an MT-PRIORITY value of 0.

Note that the MTA's Priority Assignment Policy, described above, is essentially that of the "MIXER" Priority Assignment Policy defined in Appendix B of [RFC 6710](#) -- this is a natural mapping for the MTA as its older Priority: header support was similarly based on [RFC 2156](#) (MIXER: Mapping between X.400 and RFC 822/MIME).

39.37.2 Size effects of message priority MTA options: `non_urgent_block_limit` (integer), `normal_block_limit` (integer), `second_class_block_limit` (integer), `urgent_block_limit` (integer)

Note that as of the 8.0 release, these size-based priority override MTA option effects are nullified if the MT-PRIORITY SMTP extension has been used to set an explicit priority value.

39.37.2.1 `non_urgent_block_limit`

The `non_urgent_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to lower than non-urgent priority, meaning that they will not be processed immediately and will wait for processing until the next periodic delivery run. The value is interpreted in terms of MTA blocks, as specified by the `block_size` MTA option. Note also that the `nonurgentblocklimit` channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.37.2.2 `normal_block_limit`

The `normal_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to non-urgent priority. The [Job Controller automatically pays attention to message effective processing priority](#) when scheduling delivery attempts. The value is interpreted in

terms of MTA blocks, as specified by the [block_size](#) MTA option. Note also that the [normalblocklimit](#) channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.37.2.3 `second_class_block_limit`

The `second_class_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to third class priority. The value is interpreted in terms of MTA blocks, as specified by the [block_size](#) MTA option. Note also that the [secondclassblocklimit](#) channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.37.2.4 `urgent_block_limit`

The `urgent_block_limit` MTA option may be used to instruct the MTA to downgrade the priority of messages based on size: messages above the specified size will be downgraded to normal priority. The [Job Controller automatically pays attention to message effective processing priority](#) when scheduling delivery attempts. The value is interpreted in terms of MTA blocks, as specified by the [block_size](#) MTA option. Note also that the [urgentblocklimit](#) channel option may be used to impose such downgrade thresholds on a per-channel basis.

39.38 Received header line MTA options

The MTA has a number of options related to Received: header lines and message-ids, and relating to the MTA's facility to sideline as `.HELD` files those messages that appear to be looping.

39.38.1 Syslog MTA options: `held_sndopr` (0 or 1)

The `held_sndopr` MTA option controls the production of syslog messages (on UNIX) when a message is forced into a held state due to certain suspicion thresholds. (Note that there are other potential causes of messages becoming `.HELD`, which are *not* covered by `held_sndopr` -- cases corresponding to explicit MTA administrator action such as execution of a `imsmta qclean` command, or cases where the held state can be logged as part of normal logging, such as execution of a [Sieve filter](#) `hold` action, either in an explicit Sieve filter or as a Sieve scriptlet executed due to a spam/virus filter package verdict, or application of an [address-based *_ACCESS mapping table](#) `$H` flag where syslog message generation can be performed via `$<` flag, or routing to the [hold channel](#) due to a [user status](#) or [domain status](#) of `hold`. `held_sndopr` is meant to warn of cases that might otherwise be easier to miss noticing, especially for unanticipated incoming problem messages.)

Suspicious cases where `held_sndopr` causes a syslog message include:

- A message may be forced into a held state because it has too many Received: header lines (see the various [max_*received_lines](#) MTA options for additional information):

```
HELDMSG, Header count exceeded; message has been marked .HELD automatically.
```

- Or a message may be forced into a held state because it has too many recipients (see the [holdlimit](#) channel option for more details):

HELDMSG, Max recipient count exceeded; message has been marked .HELD automatically.

- Or a message may be forced into a held state because it has too many MIME parts or levels (see the [max_mime_levels](#) and [max_mime_parts](#) MTA options):

HELDMSG, Max MIME part/level limit exceeded; message has been marked .HELD automatically.

A value of 1 for `held_sndopr` instructs the MTA to issue such messages when such cases occur. A value of 0 (the default) turns off these messages. The syslog messages will be issued using the configured [sndopr_priority](#) facility and severity.

39.38.2 Message-id: domain name MTA option: `id_domain` (string)

The `id_domain` MTA option specifies the domain name to use when constructing message IDs. If this option is not explicitly set (which is the default), then the [official host name](#) of the local channel is used.

39.38.3 Received header line MTA options: `max_local_received_lines` (integer)

As the MTA processes a message, it scans any Received: header lines attached to the message looking for references to the official local host name. (Any Received: line that the MTA inserts will contain this name). If the number of Received: lines containing this name exceeds the `max_local_received_lines` value, the message is entered into the MTA destination channel queue area in a held state. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. The default for `max_local_received_lines` is 10.

39.38.4 Received header line MTA options: `max_mr_received_lines` (integer)

As the MTA processes a message, it counts the number of MR-Received: header lines in the message's header. (MR-Received: header lines are added to messages processed by a PMDF-MR gateway.) If the number of MR-Received: lines exceeds the `max_mr_received_lines` value, the message is entered into the MTA destination channel queue area in a held state. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. The default for `max_mr_received_lines` is 20.

39.38.5 Received header line MTA options: `max_received_lines` (integer)

As the MTA processes a message, it counts the number of Received: header lines in the message's header. If the number of Received: lines exceeds the `max_received_lines` value,

max_total_received_lines
MTA option

the message is entered into the MTA destination channel queue area in a held state. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. The default for `max_received_lines` is 50.

39.38.6 Received header line MTA options:

`max_total_received_lines` (integer)

As the MTA processes a message, it counts the number of Received:, MR-Received:, X400-Received: header lines in the message's header. If the number of all such header lines exceeds the `max_total_received_lines` value, the message is entered into the MTA destination channel queue area in a held state. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. The default for `max_total_received_lines` is 100.

39.38.7 Received header line MTA options:

`max_x400_received_lines` (integer)

As the MTA processes a message, it counts the number of X400-Received: header lines in the message's header. (X400-Received: header lines are added to messages processed by a PMDF-X400, PMDF-MB400, or other X.400-to-Internet e-mail gateway.) If the number of Received: lines exceeds the `max_x400_received_lines` value, the message is entered into the MTA destination channel queue area in a held state. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. The default for `max_x400_received_lines` is 50.

39.38.8 Received header line MTA options:

`received_domain` (string)

The `received_domain` MTA option sets the domain name for the [SMTP server](#) to use when constructing Received: header lines (and Date-Warning: header lines). By default, the [official host name](#) of the local channel is used, `channel:1.official_host_name`. Note that this MTA option does not apply to the Received: header lines generated by other channels; in particular, it does not apply to Received: header lines resulting from a pass through an intermediate channel (such as a [process](#), [reprocess](#), or [conversion channel](#)). In such cases, the channel official host name---suffixed with the official host name for the local channel, in the case of short form names---will be used.

If there is no `ldap_default_domain` set, then `received_domain` is also the value that a [Sieve environment](#) "domain" will return. (If neither `ldap_default_domain` nor `received_domain` is set, then the MTA falls back to the L channel's [official host name](#).)

39.38.9 Received header line MTA options:

`received_version` (string)

RESTRICTED: Use of this option is not recommended.

The `received_version` MTA option sets the MTA version string to use when the MTA constructs Received: header lines. By default, if this option is not set, an internally constructed string is used that incorporates correct, current MTA version data.

Thus this option is the complement of the (also not recommended) [custom_version_string TCP/IP channel option](#).

39.39 Sieve filter MTA options

A number of MTA options affect the [interpretation](#) of [Sieve filters](#) and Sieve filter actions, [enable or disable optional Sieve extensions](#) or [impose limits on Sieve features or structures](#), [control the availability and operation of the Sieve duplicate extension](#), or [affect the period between repeating autoresponse messages](#), [tune caching of parsed Sieve filters](#), [enable or disable Sieve filter result logging](#), [modify error text issued in cases of Sieve filter access or processing errors](#), or [enable Sieve filter processing debugging](#).

For Sieve filter processing by external (external-to-the-MTA) components such as [imexpire](#), see also the [External filtering context MTA options](#).

39.39.1 `systemfilter` Option

The `systemfilter` MTA option takes a Sieve filter as its value, referred to as the MTA system Sieve filter. (In legacy configuration, this Sieve was stored in the `IMTA_TABLE:imta.filter` file.) Normally, this Sieve filter will be applied to every message processed by the MTA, at every enqueue; but see the [disabledestinationfilter](#) and [disablesourcefilter](#) channel options which can disable application of selected Sieve filters on a per-channel basis.

Because the MTA system Sieve filter, like channel filters, is controlled by the MTA administrator rather than by users, [certain Sieve features are available to it not normally available to users](#).

Note that because the MTA system Sieve filter is applied so frequently (normally to every message, at every enqueue), it is particularly important that it be a well-written, efficient Sieve filter.

Note that a summary of the Sieve actions performed upon a message, whether from the `systemfilter` MTA system Sieve filter or any other Sieve applying to a message, may be recorded in the MTA message transaction log file by enabling the [log_filter](#) MTA option. This is truly a summary of what was *performed*; it will not show actions that were superceded or considered but not performed.

39.39.2 Sieve filter interpretation MTA options

The MTA has several options that modify the "meaning" or "effect" (that is, the interpretation) of [Sieve filters](#):

- [decode_encoded_words](#) -- Decode encoded words during Sieve processing
- [defer_header_addition](#) -- (New in Messaging Server 7.0.5.31.0) Defer addition of headers upon [redirect action](#)
- [filter_discard](#) -- Delete discarded messages immediately *vs.* delayed
- [filter_jettison](#) -- Delete jettisoned messages immediately *vs.* delayed
- [notify_ignore_errors](#) -- Whether Sieve `notify` action invalid recipients cause an abort with an error, or are silently ignored

- [sieve_received](#) -- Sieve filters can see (an approximation of) the Received: header line that the MTA is about to add
- [sieve_redirect_add_resent](#) -- Whether Sieve redirect actions add Resent-* header lines
- [sieve_user_carryover](#) -- Whether user Sieve filters carry over when forwarding

39.39.2.1 Sieve filter interpretation MTA options: `decode_encoded_words` (integer)

RESTRICTED.

Control some decoding of encoded words during Sieve processing. The default value is 7.

39.39.2.2 Sieve filter interpretation MTA options: `defer_header_addition` (0 or 1)

The `defer_header_addition` MTA option controls whether Sieve filters see headers added to messages prior to Sieve processing. This includes, but is not limited to, headers added by list expansion processing. In older versions, such added headers would not be visible to Sieve filters. This option was added for Messaging Server 7.0u5p31 with a default of 0, meaning that added headers *are* seen by Sieve filters; setting the option to 1 restores the older behavior (where such added headers are not visible to Sieve filters on redirected messages).

Prior to the 8.0 release Sieve redirect used deferred reprocessing and thus any headers added to a redirected message were affected by this option. As of 8.0 Sieve redirect queues to the [process channel](#) and this option no longer has any effect on header visibility in redirected messages.

39.39.2.3 Sieve filter interpretation MTA options: `filter_discard` (1 or 2)

The `filter_discard` MTA option controls whether Sieve filter "discard" actions cause such discarded messages to go to the [bitbucket channel](#) (*i.e.*, be immediately discarded), or cause such messages to go to the [filter_discard channel](#) (which will leave them around for a short period before discarding them). The default is 1, meaning that messages discarded by a Sieve filter are immediately discarded. Setting this option to 2 causes discarded messages to instead be routed to the `filter_discard` channel.

39.39.2.4 Sieve filter interpretation MTA options: `filter_jettison` (1 or 2)

New in JES MS 6.1-0.01. The `filter_jettison` MTA option controls whether Sieve filter "jettison" actions cause such discarded messages to go to the [bitbucket channel](#) (*i.e.*, be immediately deleted), or cause such messages to go to the [filter_discard channel](#) (which will leave them around for a short period before deleting them). If this option is not explicitly set, it defaults to the value of the [filter_discard](#) MTA option. Since the default is `filter_discard=1`, then the default for jettisoned messages is also that such messages are immediately discarded. Setting `filter_jettison=2` (or if `filter_jettison` is not set at all, setting `filter_discard=2`) causes jettisoned messages to instead be routed to the `filter_discard` channel.

39.39.2.5 Sieve filter interpretation MTA options: `notify_ignore_errors` (0 or 1)

The `notify_ignore_errors` MTA option specifies whether specifying an invalid notification recipient address in a [Sieve "notify" action](#) will cause Sieve filter evaluation to abort with an error (0) or cause the `notify` action to be silently ignored (1). 0 is the default.

39.39.2.6 Sieve access to local Received: field `sieve_received`

[Sieve scripts](#) are necessarily evaluated after messages are received but before messages are enqueued to specific recipients. The MTA tries to include as much information in Received: fields as possible, which means deferring Received: field generation until the actual enqueue operation is performed. This means that Sieve scripts are not able to "see" the outermost Received: field that appears on the message.

Although most of the information in the outermost Received: field is available through other means, not all of it is, and even if it is available it may not be convenient to access it through other mechanisms. So the MTA normally adds reasonable approximation to the final Received: field to the message header prior to Sieve script evaluation. This addition is controlled by the `sieve_received` MTA option. A value of 1 (the default) enables the addition of this Received: field while a value of 0 disables it.

39.39.2.7 Sieve filter interpretation MTA options: `sieve_redirect_add_resent` (0 or 1)

New in 6.3-1.04. The `sieve_redirect_add_resent` MTA option sets the MTA system default for whether [Sieve "redirect" actions](#) cause addition of Resent-* header lines. The default for this option is 1, meaning to add Resent-Date:, Resent-To:, and Resent-From: header lines when performing a "redirect". This MTA system default may be overridden on a per-action basis using the `:resent` and `:noresent` arguments to "redirect". That is, setting `sieve_redirect_add_resent=1` causes these fields to be generated unless `:noresent` is used; whereas setting `sieve_redirect_add_resent=0` causes the fields to be generated only if `:resent` is used.

39.39.2.8 Sieve filter MTA options: `sieve_user_carryover` (0 or 1)

New in JES MS 6.0-0.01. The default is 0. If set to 1, [user Sieve filters](#) don't "carry over" when doing `mailDeliveryOption: forward`. This option is only relevant for direct LDAP forwarding (forwarding via `mailDeliveryOption` and `mailForwardingAddress`); it does not have any effect on other forms of forwarding.

39.39.3 Sieve filter limit MTA options

The MTA has configurable limits on how many of various [Sieve actions](#) can be applied in a Sieve filter, and on the size of certain Sieve constructs.

39.39.3.1 Sieve filter limit MTA options: `max_addheaders` (integer ≥ 0)

The `max_addheaders` MTA option sets the maximum number of [Sieve "addheader" actions](#) that can be performed in a single Sieve script. The default is 10. As of the 8.0 release, this limit only applies to user-level Sieves.

39.39.3.2 Sieve filter limit MTA options: `max_duplicates` (integer ≥ 0)

The `max_duplicates` MTA option specifies the maximum number of Sieve "duplicate" tests that may be performed by a Sieve script. The default is 2.

39.39.3.3 Sieve filter limit MTA options: `max_fileintos` (integer ≥ 0)

The `max_fileintos` MTA option specifies the maximum number of Sieve "fileinto" actions that may be performed by a Sieve script. The default is 10. As of the 8.0 release, this limit only applies to user-level Sieves.

39.39.3.4 Sieve filter limit MTA options: `max_notify`s (integer ≥ 0)

The `max_notify`s MTA option specifies the maximum number of "notify" actions that may be performed by a Sieve script. The default is 0, meaning that "notify" actions cannot be used.

New in 7.0.5, `max_notify`s is checked when processing Sieve "require" and "ihave" clauses; if such a clause is being applied on "notify", the clause will fail if `max_notify`s=0 is set.

39.39.3.5 Sieve filter limit MTA options: `max_redirect_addresses` (non-negative integer)

The `max_redirect_addresses` MTA option specifies how many addresses to read in from an external list used in a Sieve "redirect" action; additional addresses will be ignored, without an error. (See Sieve external lists for a discussion of external list use in "redirect" actions.) The default for `max_redirect_addresses` is 128.

39.39.3.6 Sieve filter limit MTA options: `max_redirects` (integer ≥ 0)

The `max_redirects` MTA option specifies the maximum number of "redirect" actions that may be performed by a Sieve script; *i.e.*, the maximum number of (Sieve script caused) forwards that may be performed. The default is 32. As of the 8.0 release, this limit only applies to user-level Sieves.

39.39.3.7 Sieve filter limit MTA options: `max_sieve_list_size` (0 < integer $\leq 10,000$)

The `max_sieve_list_size` MTA option specifies the maximum number of elements that may appear in a Sieve string-list structure (that is, strings listed within square brackets, [**(string)*]) inside a Sieve script.

The default is 64; allowed values are integers greater than 0 and less than or equal to 10000.

If a Sieve filter attempts to use more elements in a string list than this option allows, Sieve filtering will be aborted (the message being processed will be delivered normally though without Sieve filtering being applied), and the MTA will also generate a notification message

to the [Sieve owner](#) -- the [postmaster](#) for system Sieve filters ([systemfilter](#) and [channel filters](#)), as well as for Sieve filters specified on groups or lists in the [aliases file](#) or via [alias options](#), or the user whose Sieve has the error for a user's own Sieve filter. The notification message will be constructed using the [return_error.txt file](#), and will include as reason/status text "Error in sieve filter: List too large".

39.39.3.8 Sieve filter limit MTA options: `max_sieve_string_size` ($0 < \text{integer} \leq 10,000,000$)

The `max_sieve_string_size` MTA option specifies the maximum number of characters that may appear in a Sieve string. This includes both string constants, variables, and internally computed string values. The default is 65536; allowed values are integers greater than 0 and less than or equal to 10,000,000.

If a Sieve filter attempts to use more characters in a string than this option allows, Sieve filtering will be aborted (the message being processed will be delivered normally though without Sieve filtering being applied), and the MTA will also generate a notification message to the Sieve owner -- the [postmaster](#) for system Sieve filters ([systemfilter](#) and [channel filters](#)), as well as for Sieve filters specified on groups or lists in the [aliases file](#) or via [alias options](#), or the user whose Sieve has the error for a user's own Sieve filter. The notification message will be constructed using the [return_error.txt file](#).

Note that prior to the 8.0 release if a Sieve script enables variables, then Sieve Strings are further limited, with hard-coded truncation (with no error message) being performed at 8192 characters. As of 8.0, the `max_sieve_string_size` option's value is used as intended, even when variables are enabled.

39.39.3.9 Sieve filter limit MTA options: `max_vacations` ($\text{integer} \geq 0$)

The `max_vacations` MTA option specifies the maximum number of [Sieve "vacation" actions](#) that may be performed by a Sieve script. The default is 2.

Exceeding the allowed number of `vacation` actions will result in an error "Too many vacations specified" during Sieve filter evaluation.

New in 7.0.5, `max_vacations` is checked when processing Sieve "require" and "ihave" clauses; if such a clause is being applied on "vacation", the clause will fail if `max_vacations=0` is set.

39.39.3.10 Sieve filter limit MTA options: `max_variables` ($\text{integer} \geq 0$)

The `max_variables` MTA option specifies the maximum number of variables that may be used in a Sieve script. The default is 128.

Attempting to use more variables than allowed will result in an error during Sieve evaluation, "No room in table for variable: *variable-name*".

39.39.4 Sieve filter caching MTA options

At the intersection between [Sieve filter](#) processing and [MTA caching of fetched data](#), are a couple of MTA options.

39.39.4.1 LDAP lookup cache MTA options: `filter_cache_size` (integer) and `filter_cache_timeout` (integer)

The MTA maintains a per-process cache of tokenized (*i.e.*, parsed but not yet evaluated) [Sieve filters](#). This cache applies both to Sieve filters fetched from LDAP (*e.g.*, from `mailSieveRuleSource` and `mailDomainSieveRulesource` LDAP attributes, or more precisely from whatever LDAP attributes are named by the `ldap_filter` and `ldap_domain_attr_filter` MTA options), and to Sieve filters directly configured into the MTA such as [channel filters](#). The `filter_cache_size` MTA option specifies the size of this cache; the default is 500. The `filter_cache_timeout` option specifies the retention time, in seconds, for entries in this cache; the default is 600.

39.39.5 Sieve language extension MTA options

Normally, [Sieve extensions](#) are enabled in individual Sieve scripts via use of the standard Sieve `require` action; and for convenience in combining Sieve scripts, the MTA by default is lenient in permitting use of multiple `require` clauses (but see the `strict_require` MTA option). However, for a few special cases of potentially computationally "expensive" actions, the MTA permits the mail system administrator to restrict enabling these extensions. Also see the [Sieve filter limit MTA options](#), as setting the maximum allowed number of some types of action to 0 effectively disables use of that action. And see the [Sieve filter interpretation MTA options](#) which can modify how certain Sieve operations are interpreted or performed.

39.39.5.1 Sieve language extension MTA options: `enable_sieve_body` (0-2)

(New in Messaging Server 7.0u2) This option controls whether Sieve filters may use the [body extension](#). The default value is 0, meaning that body is not available. A value of 1 allows body to be used in any Sieve script. A value of 2 allows body to be used in system-level Sieves only.

39.39.5.2 Sieve language extension MTA options: `enable_sieve_ereject` (0-2)

(New in 7.2-7.02.) The [Sieve "ereject" extension](#) defined in [RFC 5429](#) is designed to be used to reject spam. Because of this it is only supposed to be available on ingress MTAs capable of returning an SMTP level error directly to a remote systems in other administrative domains. Ereject is not supposed to be available on internal MTAs incapable of sending a direct SMTP response because it's use on such systems can produce blowback spam.

The `enable_sieve_ereject` option provides the means to disable "ereject" on internal systems. A value of 1, the default, enables the use of "ereject" in all scripts. A value of 0 disables it; when `enable_sieve_ereject=0` is set, a Sieve script statement of `'require "ereject" ;'` will be ignored.

39.39.5.3 Sieve language extension MTA options: `enable_sieve_memcache` (0-2)

New in the 8.0 release. The `enable_sieve_memcache` MTA option controls whether [Sieve filters](#) may use the [memcache operator](#) to access and manipulate stored data using the memcache protocol. A value of 0 disables the use of memcache in all Sieve scripts. The default

value is 1, meaning that memcache may be used in any Sieve script. A value of 2 allows memcache to be used in system-level Sieves only.

When `enable_sieve_memcache` is set to 0, then any attempt to use memcache will result in a Sieve error "Memcache access has been disabled".

When `enable_sieve_memcache` is set to 2, then user-level attempts to use memcache will result in a Sieve error "Memcache only allowed in system-level sieves".

39.39.5.4 Sieve language extension MTA options: `enable_sieve_metermaid (0-2)`

New in the 8.0 release. The `enable_sieve_metermaid` MTA option controls whether Sieve filters may use the [metermaid operator](#) to access and manipulate data stored in [MeterMaid](#). A value of 0 disables the use of the metermaid operator in all Sieve scripts. The default value is 1, meaning that metermaid may be used in any Sieve script. A value of 2 allows metermaid to be used in system-level Sieves only.

When `enable_sieve_metermaid` is set to 0, then any attempt to use metermaid will result in a Sieve error "MeterMaid access has been disabled".

When `enable_sieve_metermaid` is set to 2, then user-level attempts to use metermaid will result in a Sieve error "MeterMaid only allowed in system-level sieves".

39.39.5.5 Sieve language extension MTA options: `enable_sieve_regex (0-2)`

The `enable_sieve_regex` MTA option controls whether Sieve filters may use the [Sieve regex extension](#) (the `:regex` match-type). A value of 0 disables the use of regex in all Sieve scripts. The default value is 1, meaning that regex may be used in any Sieve script. As of the 7 Update 2 release, a value of 2 allows regex to be used in system-level Sieves only.

When `enable_sieve_regex` is set to 2, then user-level attempts to use `:regex` will result in a Sieve error `:regex only allowed in system-level sieves`.

39.39.5.6 Sieve language extension MTA options: `strict_require (0 or 1)`

The `strict_require` MTA option controls whether or not the MTA enforces "strict" syntax rules on the location of any [require clauses](#) in Sieve filter scripts. The default is 0 (false), meaning that require clauses may appear anywhere in the Sieve script, not only at the very top of the Sieve script.

39.39.6 Sieve filter duplicate extension MTA options

As of 8.0, the MTA supports the [Sieve duplicate extension](#) specified in [RFC 7352](#).

Several MTA options relate to this support. In particular, the `duplicate_tracking_url` MTA option specifies where duplicate tracking information should be stored. At present, the value *must* be a [memcache: URL](#) of the form:

```
memcache://host:port/key-prefix
```

If the host isn't specified, it defaults to the value of the `memcache_host` MTA option. It is an error for `memcache_host` not to be set in this case. If the port isn't specified, it defaults to the value of the `memcache_port` MTA option; if that option in turn isn't specified, the default is 11211, the usual port for memcache servers. `key-prefix`, if specified, is prepended to the keys the duplicate extension sends to the memcache server.

Note that duplicate tests are performed during Sieve evaluation but no memcache updates are performed. It is only after the message has been successfully processed that updates are done.

Also note that duplicate information is implicitly qualified by the owner of the Sieve. In the case of [system-level Sieves](#), this will be the applicable postmaster address, so system-level Sieves operate in shared namespace(s). Note that the `:handle` argument can be used to force system-level Sieves to operate in their own namespace.

39.39.6.1 Duplicate test storage timeout minimum: `duplicate_minimum_timeout` (integer)

(New in 8.0.) The `duplicate_minimum_timeout` MTA option establishes a minimum value, in seconds, for the Sieve `"duplicate" ":seconds"` parameter that controls how long test information is retained. Values lower than the minimum are silently adjusted up to the minimum; no error occurs. The default value for `duplicate_minimum_timeout` is 0.

39.39.6.2 Duplicate test storage timeout maximum: `duplicate_maximum_timeout` (integer)

(New in 8.0.) The `duplicate_maximum_timeout` MTA option establishes a maximum value, in seconds, for the Sieve `"duplicate" ":seconds"` parameter that controls how long test information is retained. Values higher than the maximum are silently adjusted down to the maximum; no error occurs. The default value for `duplicate_maximum_timeout` is 604800 seconds (7 days).

39.39.6.3 Duplicate test storage timeout default: `duplicate_timeout_default` (non-negative integer)

The `duplicate_timeout_default` MTA option specifies the default timeout, in seconds, for storage of information provided to the Sieve `"duplicate" test`. The default is 604800 seconds (seven days).

39.39.6.4 Sieve duplicate detection: `duplicate_tracking_url` (memcache URL)

The `duplicate_tracking_url` MTA option specifies where duplicate tracking information produced by the [Sieve duplicate extension](#) should be stored. At present the value must be a `memcache: URL` of the form:

```
memcache://host:port/key-prefix
```

If the host isn't specified, it defaults to the value of the [memcache_host](#) MTA option. It is an error for [memcache_host](#) not to be set in this case.

If the port isn't specified it defaults to the value of the [memcache_port](#) MTA option; if that option in turn isn't specified the default is 11211, the usual port for memcache servers.

`key-prefix`, if specified, is prepended to the keys the duplicate extension sends to the memcache server.

39.39.6.5 Sieve filter limit MTA options: `max_duplicates` (integer ≥ 0)

The `max_duplicates` MTA option specifies the maximum number of Sieve "duplicate" tests that may be performed by a Sieve script. The default is 2.

39.39.7 Sieve filter error text MTA options

A few of the `error_text_*` MTA options discussed in [error_text MTA options](#) relate specifically to Sieve filter error messages; see the [error_text_sieve_access](#), [error_text_sieve_syntax](#), [error_text_source_sieve_access](#), [error_text_source_sieve_syntax](#), and [error_text_sieve_authorization](#) MTA options.

39.39.8 Sieve filter log and debug MTA options

A few MTA options relate to logging of Sieve filter applied actions, and debugging of MTA Sieve filter processing.

For debugging of Sieve filters, see also the [mm_debug](#) MTA option, the `test -expression` utility, and the [Sieve debug action](#).

39.39.8.1 Debug MTA options: `filter_debug` (0 or 1)

New in Messaging Server 6.2. Control whether the stack state information part of [Sieve filter](#) debugging is put in debug logs. (Note that this is quite "low level" debugging, not likely to be of interest unless requested by Oracle support.)

For debugging of Sieve filters, see also the [imsimta test -expression utility](#) and the [mm_debug](#) MTA option along with the [Sieve debug action](#).

39.39.8.2 Transaction logging MTA options: `log_filter` (0-3)

The `log_filter` option controls whether or not any mailbox filter actions (Sieve filter actions) applicable to the message are logged in enqueue "E" records or included in [LOG_ACTION mapping](#) probes. Bit 0 (value 1), if set, causes filter information to appear in log entries. New in 7.0-3.01, bit 1 (value 2), if set, causes filter information to be included in [LOG_ACTION mapping](#) probes. This information appears after the optional "intermediate" and "original" forms of the destination address (see the [log_intermediate](#) MTA option), before the [SMTP diagnostic field](#) (which itself only appears for SMTP messages). In XML-compatible format (`log_format` set to 4), Sieve filter action(s) logging, if enabled, appears as the `f1` attribute. The filter action(s) will be enclosed within single quote characters. (In

"D" records, one sees merely the two single quote characters, as no filter action is applicable during dequeue.) As of JES MS 6.3p1, enabling XML format (`log_format` set to 4) causes the default for `log_filter` to be 1 (Sieve filter action logging enabled); with any other format, the default is 0, (Sieve filter actions are not logged), as in previous versions. With `log_filter` set to 1, one might see, for instance

```
'fileinto "SPAM"'
```

or

```
'redirect "user@domain.com"'
```

As of 8.0, a "warn" clause may also be present; see the discussion in [Sieve warn extension](#).

Note that the case of a "reject" action is special, due to the inherent nature of the "reject" action. In this case, what occurs is the enqueue of a new message (a [Message Disposition Notification](#)) by the original enqueueing channel to the [process channel](#), and that new message has an implicit keep occurring. As there is no enqueue of the rejected message, the "reject" does not show up in the filter action field of any transaction log record.

As of MS 8.0, the maximum size of the `log_filter` field (the maximum length of the string recording what Sieve actions were applied) has been increased from 256 to 1024 characters.

39.39.8.3 Transaction logging MTA options: `log_transactionlog` (0-3)

(New in 8.0.) The `log_transactionlog` MTA option controls whether [Sieve transactionlog action](#) strings are included in [MTA message transaction log records](#). The option defaults to 0, meaning that such Sieve actions are not logged. Setting bit 0 (value 1) causes the `transactionlog` string to be logged at the very end of enqueue ("E") records. The XML attribute name in XML format logs (`log_format=4`) is "t1". Setting bit 1 (value 2) causes the the `transactionlog` string to be included in the [LOG_ACTION mapping table](#) probe, again at the very end.

For instance, with MTA message transaction logging enabled (the `logging` channel option set for all channels) and with `log_transactionlog` also set:

```
msconfig> show logging
role.channel.defaults.logging
msconfig> set log_transactionlog 1
```

then an MTA system filter (see for instance the `msconfig` command `edit filter`) including:

```
require "variables";
if header :matches "subject" "*" {transactionlog "${0}";}
```

will cause MTA message transaction log records to include the contents of the Subject: header line. (Note that merely logging the Subject: header line of messages passing through the MTA could instead be achieved via use of [log_header](#) or [logheader](#). But the Sieve

script approach allows more fine-tuning, as a Sieve script can be coded with complex logic dependent upon other message details, such as message sender or recipient, presence of specific strings in the header, *etc.*)

39.40 Spamfilter MTA options

The MTA has a number of options affecting Brightmail, ClamAV, Cloudmark, ICAP, Milster (as of JES MS 6.3), SpamAssassin, Symantec SAVSE or similar virus/spam filtering plug-in facilities. These options are also used to integrate with AXS:One archiving. Most of these options affect the operation of the spam/virus filter package or archive package itself; however, there are also options that affect which user(s) and domains(s) get such filtering or archiving applied, such as the [ldap_optinN](#), [ldap_source_optinN](#), and [ldap_domain_attr_optinN](#) MTA options.

For each virus/spam filtering package used, typically at a minimum a pair of options [spamfilterN_library](#) and [spamfilterN_config_file](#) must be specified, telling the MTA the location of a library of callable routines for that virus/spam filter package and providing the location of a configuration file containing some configuration information specific to that virus/spam filter package. Additional [spamfilter*](#) MTA options may be used to further fine-tune the MTA's utilization of the virus/spam filter package.

Prior to JES MS 6.2, the MTA supported use of one spam/virus filter package callout (of the site's choice). In JES MS 6.2, use of up to four spam/virus filter packages callouts was supported (via [spamfilter1_*](#) through [spamfilter4_*](#) options). New in JES 5 (JES MS 6.3), up to eight spam/virus filter package callouts are supported (via [spamfilter1_*](#) through [spamfilter8_*](#) options).

The spam/virus filter packages get called asynchronously, being passed the original* (not yet processed by the MTA) message; thus multiple spam/filter packages may be working on the same message in parallel. The spam/filter packages report their verdicts back to the MTA on their own timelines, as they respectively finish their work. The verdicts from the spam/virus filter packages are converted by the MTA into [Sieve scriptlets](#), per the MTA's [spamfilterN_verdict_M](#), [spamfilterN_action_M](#) option pairs as well as the default cases controlled via the [spamfilterN_null_action](#) and [spamfilterN_string_action](#) MTA options. The MTA then evaluates these Sieve scriptlets in order from spam/filter package 1 to spam/filter package 8; this in particular means that higher numbered, "later" spam/filter packages can see the results, such as added headers, from lower numbered, "earlier" spam/filter packages and potentially [override](#) them, if desired, although in regards to combining/weighting/overriding the verdicts-and-effects of multiple spam/virus filter packages, a flexible and perhaps more straightforward approach is to convert the spam/virus filter package verdicts into various added header lines via configuration of the above MTA options and then have the MTA's [systemfilter](#) see, consider, and make decisions based upon all the results (all the added header lines).

* Regarding the passing of the "original" (as received) message to the spam/virus filter packages, a few modifications are configurable via the [spamfilterN_final](#), [spamfilterN_includeheaders](#), [spamfilterN_received](#), and [spamfilterN_returnpath](#) MTA options.

For Spam/virus filter package invocation by external (external-to-the-MTA) components such as [imexpire](#), see also the [External filtering context MTA options](#),

See also the [access_errors](#) MTA option which affects the error text used when a spam/virus filter package rejects a recipient address.

39.40.1 Spamfilter MTA options: optin_user_carryover (bitmask)

New in JES MS 6.2. The `optin_user_carryover` MTA option controls whether user spam/virus filter "opt in" requests will "carry over" when doing forwarding. That is, if the original recipient has opted-in but has then forwarded their e-mail to some other recipient, does that other recipient get the "opt in" effect?

Bit 0 (value 1): setting this bit means that "opt in" effect is disabled for all forwarded-to address(es). Bit 1 (value 2) controls the behavior for [domain "opt in"](#); setting the bit disables the "opt in" effect for the forwarded-to address(es). Bit 2 (value 4) means that [user "opt in"](#) overrides any previous user/domain "opt in" setting. Bit 3 (value 8) controls the behavior for aliases (typically lists) marked with the [alias_optin](#) alias option or [named parameter \[OPTIN\]](#); setting the bit disables the "opt in" effect for the forwarded-to address(es). The default is 0. Note that this option applies globally to *all* spam/virus filter packages; it does *not* come in numbered variants to apply only to one spam/virus filter package or another.

39.40.2 Spamfilter MTA options: spamfilterN_library (filepath)

The `spamfilterN_library` MTA options specify where a spam/virus filter package library image is located. The default is to set no library.

For instance, when using Brightmail, one of these options would be set to point to the path to the `libbmiclient.so` library image, *e.g.*,

```
msconfig> set mta.spamfilter1_library /opt/mailwall/lib/libbmiclient.so
```

Or for instance, when using SpamAssassin, one of these options would be set to point to the path to the `libspamass.so` library image, *e.g.*,

```
msconfig> set mta.spamfilter2_library /opt/SUNWmsgsr/lib/libspamass.so
```

or if using an MTA "logical name" for the directory specification:

```
msconfig> set mta.spamfilter2_library IMTA_TABLE:libspamass.so
```

Or for instance, when using a Milter, one of these options would be set to point to the path to the `libmilter.so` library image, *e.g.*,

```
msconfig> set mta.spamfilter3_library /opt/SUNWmsgsr/lib/libmilter.so
```

Or for instance, when using ICAP, one of these options would be set to point to the path to the `libicap.so` library image, *e.g.*,

```
msconfig> set mta.spamfilter4_library /opt/SUNWmsgsr/lib/libicap.so
```

39.40.3 Spamfilter MTA options: **spamfilterN_config_file (filepath)**

The `spamfilterN_config_file` MTA options are used specify the location of the configuration file for a spam/virus filter package. The value of this option is passed to the spam/virus filter package for it to use to locate its configuration file; thus note that MTA "logical names" (e.g., `IMTA_TABLE`) should not be used; and whether the spam/virus filter package prefers absolute file paths (including full directory path), or prefers a bare filename (presumably located in some fixed/default directory) can vary with the specific spam/virus filter package in use.

This option is specifying the location of a configuration file *for the spam/virus filter plug-in itself*; the MTA passes this option value (the location of this configuration file) to the spam/virus filter package and it is then up to the spam/virus filter package to open and read that specified configuration file. Thus in particular the actual options available and supported within the specified file are dependent upon which spam/virus filter package is accessing its (own) configuration file. For details on what options may be specified within a spam/virus filter package's own configuration file, see, respectively:

- [BrightmailspamfilterN_config_file](#)
- [ClamAV spamfilterN_config_file](#)
- [ICAP spamfilterN_config_file](#)
- [Milter spamfilterN_config_file](#)
- [SpamAssassin spamfilterN_config_file](#)
- [Archive spamfilterN_config_file](#)

39.40.4 Spamfilter MTA options: **spamfilterN_null_optin (string)**

Normally, the simple presence of a spam filter `optin` attribute (the attribute named by the `ldap_optinN` MTA option or the attribute named by the `ldap_source_optinN` MTA option in a user entry, or the attribute named by the `ldap_domain_attr_optinN` MTA option in a domain entry) turns on filtering; all the value determines is what sort of filtering will be done. This isn't compatible with some directory maintenance and provisioning tools that cannot easily delete an attribute, that always provide the attribute but assume some sort of "off" or "null" value for the attribute is available that doesn't enable filtering. The `spamfilterN_null_optin` MTA options allow for better interaction with such directory tools. A `spamfilterN_null_optin` MTA option specifies what value a spam filter `optin` attribute must have to be ignored (by spam/virus filter package *N*). The default value for these options is the empty string, which means that by default a present but empty `optin` attribute is ignored.

39.40.5 Spamfilter MTA options: **spamfilterN_action_M (URL)** **spamfilterN_verdict_M (string)**

Each pair of options `spamfilterN_verdict_M` and `spamfilterN_action_M` for particular values of *N* and *M* specifies the the action that the MTA should take upon receiving the corresponding (*M*) verdict from the virus/spam filter package number *N*. *N* can range

between 1 and 4 as of JES MS 6.2 and range between 1 and 8 as of JES MS 6.3; that is, up to four (as of JES MS 6.2) or eight (as of JES MS 6.3) virus/spam filter packages may be in use simultaneously. *M* can range between 0 and 7; that is, up to eight such pairs may be specified per virus/spam filter package.

In legacy configuration, the `spamfilter_action_M` and `spamfilter_verdict_M` MTA options were synonyms, respectively, for `spamfilter1_action_M` and `spamfilter1_verdict_M` MTA options; Unified Configuration does not support those old aliases other than for upgrade purposes.

Each `spamfilterN_verdict_M` MTA option specifies a possible verdict string from a virus/spam filter package such as Brightmail, with *N* corresponding to the *N* in [spamfilterN_library](#) and [spamfilterN_config_file](#); that is, *N* specifies the (arbitrary) "number" identifying the particular virus/spam filter package. New in JES MS 6.2p1, the `*_verdict_*` MTA option value may contain wildcards and glob matches; the MTA will do pattern matching on the verdict string returned by a spam/virus filter package. See the table of [Mapping pattern wildcards](#) for the sorts of wildcards and globs that may be used in the `*_verdict_*` MTA option settings; (note that "saving" of wildcards or globs is disabled in this context, so in particular one may use more than ten wildcards or globs in such a value). New in JES MS 6.3, the length of string argument to each such option has been increased to 1024 characters (where previously the limit was 256 characters).

For each such option (verdict), a corresponding `spamfilterN_action_M` MTA option should also be set, specifying what action to take when the corresponding verdict is returned. The value of a `spamfilterN_action_M` MTA option should be a [URL](#) that resolves to a [Sieve filter](#) (where the Sieve filter specifies the action to take). The URLs can use [MTA URL substitution sequences](#), as in [Table of LDAP URL substitution sequences](#), though most such substitutions have no meaning or are irrelevant in this particular context, and in a few cases the meanings are different in this context. In particular:

Table 39.32 spamfilterN_action_M MTA option values

Substitution sequence	Description
<code>\$(U)</code>	The verdict name string
<code>\$(A)</code>	The address that this verdict is associated with
<code>\$(M)</code>	(New in JES MS 6.0) The detailed verdict string (if the spam/virus filter package provided one - not all do)

In JES MS 6.2, *N* can range between 1 and 4; that is, up to four virus/spam filter packages may be in use simultaneously. As of JES 5 (JES MS 6.3), *N* can range between 1 and 8; that is, up to eight virus/spam filter packages may be in use simultaneously. *M* can range between 0 and 9; that is, up to ten such pairs may be specified per virus/spam filter package.

For instance, if the spam/virus filter package configured in the MTA as package 1 might return as its "verdict" a string of either the form

```
spam...arbitrary-text...X-HEADER: SPAM...more-text...
```

or

```
spam...arbitrary-text...discard...more-text...
```

then one might configure MTA options as follows in legacy configuration:

```
SPAMFILTER1_VERDICT_1=spam*X-Header: SPAM*
SPAMFILTER1_ACTION_1=data:, addheader "X-Header" "SPAM";
SPAMFILTER1_VERDICT_2=spam*discard*
SPAMFILTER1_ACTION_2=data:, discard;
```

or in Unified Configuration:

```
msconfig> set mta.spamfilter1_verdict_1 "spam*X-Header: SPAM*"
msconfig# set mta.spamfilter1_action_1 'data:, addheader "X-Header" "SPAM";'
msconfig# set mta.spamfilter1_verdict_2 spam*discard*
msconfig# set mta.spamfilter1_action_2 "data:, discard;"
msconfig# show spamfilter1*
role.mta.spamfilter1_action_1 = data:, addheader "X-Header" "SPAM";
role.mta.spamfilter1_verdict_1 = spam*X-Header: SPAM*
role.mta.spamfilter1_verdict_2 = spam*discard*
role.mta.spamfilter1_action_2 = data:, discard;
```

Note The special value "data: , \$M" for a spamfilterN_action_M MTA option has some special, short-circuited handling, in that it causes the verdict to be used literally as a Sieve scriptlet itself, omitting the usual URL expansion processing. For "short" verdicts, this is a difference that makes no difference, but it avoids the length limitations imposed during URL expansion that could potentially become relevant for longer verdicts---such as those that [milter](#) likes to return. Note that other uses of \$M in a setting still result in substitution of the original verdict string but *with* normal URL expansion (thus subject to length limits); it is only the special, literal setting "data: , \$M" that gets the special, short-circuited handling.

Note: Brightmail has a concept of a "default" verdict, intended to mean merely "deliver normally". Brightmail is typically configured so that a Brightmail "clear of any virus or spam" result is set to a Brightmail "destination" (a "verdict" in the MTA's terminology) of "inbox", with "inbox" also being set to be the "default destination". In older Brightmail configuration, this Brightmail configuration of "default destination" would be something like:

```
blSWCClientDestinationdefault: inbox
```

The MTA has support for Brightmail's default destination concept, implemented by the MTA checking whether a verdict that it has received from Brightmail is Brightmail's default destination and if so, the MTA forcibly performs a plain "keep" Sieve action (forces delivery to the Inbox) and *does not apply* any spamfilterN_verdict/spamfilterN_action processing. Thus to achieve some other result for Brightmail clear messages, that is, to get spamfilterN_verdict/spamfilterN_action processing to occur for "clear" messages (such as perhaps, adding a header line saying that the message was cleared by Brightmail as well as delivering to the Inbox), Brightmail must be configured differently: either configure Brightmail so that "inbox" is *not* Brightmail's "default destination", or configure Brightmail to return some destination (some verdict, in MTA terminology) of other than "inbox". Either way, the MTA must see a non-default (in Brightmail's opinion) destination/verdict in order for it to apply "normal" spamfilterN_verdict/spamfilterN_action processing.

In addition to the explicitly-specified-verdict/corresponding-action pairs discussed above, there are also two additional types of MTA option that specify the behavior of the MTA

when other verdicts are returned: the [spamfilterN_null_action](#) options controlling behavior when a so-called "null" verdict is returned, and the [spamfilterN_string_action](#) options controlling behavior when an unrecognized verdict (a verdict without a matching `spamfilterNverdict_M` value) is returned.

39.40.6 Spamfilter MTA options: `spamfilterN_final` (bitmask)

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 respective `spamfilterN_final` ($N=1-8$) MTA option. 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 of local users. If bit 0 (value 1) of `spamfilterX_final` 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 use in subsequent forwarding operations. The `spamfilterN_final` MTA options are only available in JES MS 6.0 and later; iMS 5.2 behaves as if the option had the default value of 0.

In JES MS 6.2 and later, bit 1 (value 2) of `spamfilterN_final` 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.

In 7.3-11.01 and later bit 2 (value 4) of `spamfilterN_final`, if set, causes the "initial" address to be passed to the spam filter. This is the address that was initially passed to the alias expansion process. Bit 1 can be used to strip source routes from such addresses if desired.

39.40.7 Spamfilter MTA options: `spamfilterN_includeheaders` (0 or 1)

(New in 7.0.5) Each `spamfilterN_includeheaders` MTA option specifies whether or not any header lines added to a message due to a [\\$A flag](#) in either a [FROM_ACCESS mapping table](#) or a [recipient address *_ACCESS mapping table](#) will be passed over to the respective N th spam/virus filter package as part of the regular message header. A value of 1 causes such header lines to be passed over; a value of 0, the default, disables this capability.

39.40.8 Spamfilter MTA options: `spamfilterN_null_action` (URL)

The `spamfilterN_null_action` MTA options specify, respectively, the Sieve action to take when a "null" verdict is returned by the N th spam/virus filter package. The default value for these options is:

```
data: ,discard;
```

meaning that a null verdict is interpreted as a request to discard the message.

39.40.9 Spamfilter MTA options: `spamfilterN_received` (0-7)

Some spam/virus filter packages operate best if provided with the "current" Received: header line -- the sort that the MTA will be generating for actual prefixing of the message, but subsequent to the spam/virus filter package scanning. A `spamfilterN_received` MTA option controls whether the MTA generates a most recent Received: header line to pass to the *N*th spam/virus filter package. This is a bit-encoded value, but any nonzero value specifies that the MTA *does* pass a pseudo-Received: header line to the spam/virus filter package. Note that this pseudo-Received: header line is not necessarily *exactly* what the MTA will truly end up inserting, but it does contain the same routing information (in particular client source IP address) which tends to be the item of especial interest to spam/virus filter packages. SpamAssassin in particular tends to operate better when provided with such a Received: header line. Bit 1 (value 0) in the option is used to cause the header to be generated without any additional options. New in Messaging Server 7.0.5 is support for the bit 1 (value 2), which means to pass the spam/virus filter package a synthesized Received: header line that includes an additional clause:

```
(envelope-sender mail-from-address)
```

Some SpamAssassin configurations use such a clause in the Received: header line as the source of MAIL FROM addresses instead of using the standards-compliant Return-path: field.

As of the 8.0.1 release, bit 2 (value 4) is used to specify that the header *NOT* be generated during reprocessing operations. (This is useful since the enqueue to the [reprocess channel](#) already added a Received: header field.)

39.40.10 Spamfilter MTA options: `spamfilterN_returnpath` (0 or 1)

The `spamfilterN_returnpath` ($N=1-8$) MTA options control whether or not a synthesized Return-path: field is prepended to the message passed to the associated spam filter. (Return-path: fields are normally only added during final delivery; however, some spam filters may only be able to process envelope From address information if it is provided in a Return-path: field.) A nonzero value causes the field to be inserted. The default value is 0.

These options are new in 7.0-3.01 and aren't available in previous versions, which never insert the field.

39.40.11 Spamfilter MTA options: `spamfilterN_string_action` (URL)

The `spamfilterN_string_action` MTA options specify the default [Sieve filter](#) actions to apply whenever the correspondingly numbered spam/virus filter plugin returns a verdict string that does not have an explicit corresponding action set. That is, specify the Sieve filter actions to apply whenever a verdict string is returned by plugin *N* that does not have a `spamfilterN_verdict_M` match for any *M*.

The default for the `spamfilterN_string_action` MTA options is:

```
data:, require "fileinto"; fileinto "$U";
```

Prior to JES MS 6.2p8, the (unexpanded) string specified as the value for such an option was limited to 256 characters (with truncation occurring if it was longer); as of JES MS 6.2p8, the limit is 1024 characters. The length of the string *after* any expansions are performed has been 1024 characters since at least JES MS 6.1.

When using a Militer, the `spamfilterN_string_action` option *must* be set to:

```
data:,$M
```

So for instance:

```
msconfig> show spamfilter3_*
role.mta.spamfilter3_config_file = /opt/sun/comms/messaging64/config/miltertest.dat
role.mta.spamfilter3_library = /opt/sun/comms/messaging64/lib/libmilter.so
msconfig> set spamfilter3_string_action "data:,$M"
```

This setting is using the `$M` substitution (see the discussion of such substitutions in the discussion of the `spamfilter1_action_0` MTA option) which means to use the detailed verdict string provided by the militer.

39.41 SPF MTA options

A number of MTA options exist affecting SPF lookups. Note that SPF lookups are enabled via channel options such as `spfhello`; the MTA options affect the interpretation of SPF results and errors. See also the [SPF_LOCAL mapping table](#), which may be used to avoid performing actual DNS lookups for selected (typically local) domains. See also the [SRS MTA options](#), as many sites using SPF will also want to enable SRS address encoding.

Sender Policy Framework, or SPF, formerly referred to as Sender Permitted From, is a mechanism that attempts to prevent email forgery. It works 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. MTA support for SPF was implemented in JES MS 6.3-0.15.

See also the `error_text_spf_*` MTA options which configure the error text issued in cases of SPF errors.

39.41.1 SPF MTA options: `spf_smtp_status_fail` (2, 4, or 5)

The `spf_smtp_status_fail` MTA option controls whether SPF Fail results are ignored (considered as successes), interpreted as temporary failures, or (the default) interpreted as permanent failures, with values of 2, 4, or (the default) 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of SPF lookups that succeeded in finding a relevant SPF record in the DNS and where the SPF record specified an explicit Fail for the *exact* domain name being checked; for interpretation of SPF lookups that succeed in finding an explicit Fail all SPF record applying to subdomains including the domain name being checked, see instead the `spf_smtp_status_fail_all` MTA option; and for interpretation of DNS-level errors in the SPF lookup attempt, see instead the `spf_smtp_status_permerror` MTA option.)

39.41.2 SPF MTA options: `spf_smtp_status_fail_all` (2, 4, or 5)

The `spf_smtp_status_fail_all` MTA option controls whether SPF Fail "all" results are ignored (considered as successes), interpreted as temporary failures, or (the default) interpreted as permanent failures, with values of 2, 4, or (the default) 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of SPF lookups that succeeded in finding a relevant SPF record in the DNS and where the SPF record specified an explicit Fail for all subdomains of some domain name where the domain name being checked matched; for interpretation of SPF lookups that find a relevant SPF record for the *exact* domain name being checked, see instead the [spf_smtp_status_fail](#) MTA option; and for interpretation of DNS-level errors in the SPF lookup attempt, see instead the [spf_smtp_status_permerror](#) MTA option.)

39.41.3 SPF MTA options: `spf_smtp_status_permerror` (2, 4, or 5)

The `spf_smtp_status_permerror` MTA option controls whether DNS permanent errors attempting SPF lookups are ignored (considered as successes), interpreted as temporary failures, or (the default) interpreted as permanent failures, with values of 2, 4, or (the default) 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of DNS level errors in the attempted SPF lookups, not the interpretation of "failed to verify" results from an SPF lookup; for that, instead see the [spf_smtp_status_fail](#) and [spf_smtp_status_fail_all](#) MTA options.) The default value is 5, meaning that such DNS level errors are considered to correspond to permanent SPF failures and result in rejection of the message.

The point in the SMTP dialogue at which the SPF lookup is attempted, hence at which the DNS error is encountered, will influence what error is returned; see the [spfhello](#), [spfmailfrom](#), and [spfrcptto](#) channel options. So with `spfhello` set on an incoming channel, if the SPF lookup of the domain specified on the client's HELO or EHLO command encounters a permanent DNS error, then with `spf_smtp_status_permerror=5` set, the SMTP server would issue a permanent rejection:

```
500 5.5.2 Permanent error in SPF verification of HELO domain
```

whereas with `spf_smtp_status_permerror=4` set, the SMTP server would instead issue a temporary rejection:

```
451 4.4.3 Permanent error in SPF verification of HELO domain
```

At the MAIL FROM: and RCPT TO: stages of the SMTP dialogue, the error text also is configurable via the [error_text_spf_permerror_5](#) and [error_text_spf_permerror_4](#) MTA options. So with `spfmailfrom` or `spfrcptto` set on an incoming channel, if the SPF lookup of the domain from the MAIL FROM: command encounters a permanent DNS error, then with `spf_smtp_status_permerror=5` set the SMTP server would issue a permanent rejection (default text):

```
550 5.5.0 permanent error in SPF verification of MAIL FROM domain (domain-name)
```

spf_smtp_status_softfail
MTA option

or using whatever text is configured via the [error_text_spf_permerror_5](#) MTA option:

```
550 5.5.0 error_text_spf_permerror_5
```

whereas with `spf_smtp_status_permerror=4` set, such an error would result in merely a temporary rejection at the MAIL FROM: stage (`spfmailfrom`) such as (default text):

```
450 4.5.1 permanent error in SPF verification of MAIL FROM domain (domain-name)
```

or at the RCPT TO: stage (`spfrcptto`) such as:

```
452 4.5.1 permanent error in SPF verification of MAIL FROM domain (domain-name)
```

or using whatever error text is explicitly configured via the [error_text_spf_permerror_4](#) MTA option, hence at the MAIL FROM: stage:

```
450 4.5.1 error_text_spf_permerror_4
```

or at the RCPT TO: stage:

```
452 4.5.1 error_text_spf_permerror_4
```

39.41.4 SPF MTA options: `spf_smtp_status_softfail` (2, 4, or 5)

The `spf_smtp_status_softfail` MTA option controls whether SPF SoftFail results are ignored (considered as successes), interpreted as temporary failures, or interpreted as permanent failures, with values of 2 (the default), 4, or 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of SPF lookups that succeeded in finding a relevant SPF record in the DNS and where the SPF record specified an explicit SoftFail for the exact domain name being checked. For the case of a SoftFail "all" SPF record that matched the domain name in a more wildcarded way, see instead the [spf_smtp_status_softfail_all](#) MTA option. Or for the interpretation of DNS-level temporary errors in the SPF lookup attempt, see instead the [spf_smtp_status_temperror](#) MTA option.)

39.41.5 SPF MTA options: `spf_smtp_status_softfail_all` (2, 4, or 5)

The `spf_smtp_status_softfail_all` MTA option controls whether SPF SoftFail "all" results are ignored (considered as successes), interpreted as temporary failures, or interpreted as permanent failures, with values of 2 (the default), 4, or 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of SPF lookups that succeeded in finding a relevant SPF record in the DNS and where the SPF record specified an explicit SoftFail all subdomains of some domain name where the domain name being checked matched; for interpretation of SPF lookups that find a relevant SPF record for the *exact* domain name being checked, see instead the [spf_smtp_status_softfail](#) MTA option. And for interpretation of DNS-level temporary errors in the SPF lookup attempt, see instead the [spf_smtp_status_temperror](#) MTA option.)

39.41.6 SPF MTA options: `spf_smtp_status_temperror` (2, 4, or 5)

The `spf_smtp_status_temperror` MTA option controls whether DNS temporary errors attempting SPF lookups are ignored (considered as successes), interpreted as temporary failures (the default), or interpreted as permanent failures, with values of 2, 4 (the default), or 5, respectively, controlling this. (To emphasize, this option is controlling the interpretation of DNS level temporary errors during the attempted SPF lookups, *not* the interpretation of "soft failure" results from an SPF lookup; for that, instead see the `spf_smtp_status_softfail` and `spf_smtp_status_softfail_all` MTA options.)

The default value is 4, meaning that such DNS level temporary errors are considered to correspond to temporary SPF failures and result in temporary rejections (deferrals) of the message. The point in the SMTP dialogue at which the SPF lookup is attempted, hence at which the DNS error is encountered, will influence what error is returned. See the `spfhello`, `spfmailfrom`, and `spfrcptto` channel options. So with `spfhello` set on an incoming channel, if the SPF lookup of the domain specified on the client's HELO or EHLO command encounters a temporary DNS error, then with `spf_smtp_status_temperror=5` set the SMTP server would issue a permanent rejection:

500 5.5.2 Temporary error in SPF verification of HELO domain

whereas with `spf_smtp_status_temperror=4` set (the default), the SMTP server would instead issue a temporary rejection:

451 4.4.3 Temporary error in SPF verification of HELO domain

At the MAIL FROM: and RCPT TO: stages of the SMTP dialogue, the error text also is configurable via the `error_text_spf_temperror_5` and `error_text_spf_temperror_4` MTA options. So with `spfmailfrom` or `spfrcptto` set on an incoming channel, if the SPF lookup of the domain in the MAIL FROM: command encounters a temporary DNS error, then with `spf_smtp_status_temperror=5` set the SMTP server would issue a permanent rejection (default text):

550 5.5.0 temporary error in SPF verification of MAIL FROM domain (*domain*)

or using whatever text is configured via the `error_text_spf_temperror_5` MTA option:

550 5.5.0 `error_text_spf_temperror_5`

whereas with `spf_smtp_status_temperror=4` set (that is, the default) such an error would result in merely a temporary rejection at the MAIL FROM: stage (`spfmailfrom`) such as (default text):

450 4.5.1 temporary error in SPF verification of MAIL FROM domain (*domain*)

or at the RCPT TO: stage (`spfrcptto`) such as:

452 4.5.1 temporary error in SPF verification of MAIL FROM domain (*domain*)

`spf_max_dns_queries` MTA
option

or using whatever error text is explicitly configured via the [`error_text_spf_temperror_4`](#) MTA option, hence at the MAIL FROM: stage:

450 4.5.1 `error_text_spf_temperror_4`

or at the RCPT TO: stage:

452 4.5.1 `error_text_spf_temperror_4`

39.41.7 SPF MTA options: `spf_max_dns_queries` (integer)

The `spf_max_dns_queries` MTA option specifies the maximum number of DNS queries per SPF check. The default is 10, which accords with the requirement in Section 10.1 of [RFC 4408 \(SPF\)](#). (Setting this option to a value above 10 thus violates the [RFC 4408](#) requirement.)

39.41.8 SPF MTA options: `spf_max_recursion` (integer)

The `spf_max_recursion` MTA option's default is 10.

39.41.9 SPF MTA options: `spf_max_time` (integer)

The `spf_max_time` MTA option specifies the maximum amount of time, in seconds, permitted when performing an SPF check. If an SPF check does not complete in this amount of time, an SPF TempError will be returned. The default is 45. ([RFC 4408 \(SPF\)](#) in Section 10.1 recommends allowing at least 20 seconds.)

39.42 SRS MTA options

The MTA has a number of options relating to SRS (Sender Rewriting Scheme). SRS is a mechanism that can solve certain forwarding problems inherent in SPF (Sender Policy Framework, formerly referred to as Sender Permitted From). For a discussion of SPF itself, see [SPF MTA options](#) and the [spf * channel options](#).

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 of course 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 one's 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 is new in JES MS 6.3-1.04. SRS address decoding is enabled by setting the [srs_domain](#) and [srs_secrets](#) MTA options; setting the [srs_maxage](#) MTA option is optional as it has a reasonable default value. See the discussions of the specific options for more details.

Note: 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.

Enabling SRS address encoding must be more precisely configured. In particular, it should only be done to envelope From addresses that you *know* are associated with forwarding activity. In addition to requiring that the SRS domain be configured via [srs_domain](#) and that the decoding keys be set via the [srs_secrets](#) MTA option already mentioned, additional configuration is required via the [*srs](#) channel options, controlling exactly which addresses, on exactly which messages, have SRS encoding applied.

Prior to the 8.0 release, note that SRS decoding of addresses, as for notification messages routing back through the SRS MTA, could run afoul of the MTA's normal "relay blocking" configuration. In particular, for a "typical" configuration where all three [*srs](#) channel options are set on the [tcp_local](#) channel, this would be an issue. See [SRS and Relay Blocking](#) for a work around approach.

See also the [error_text_srs_*](#) MTA options, which control the exact error text issued when SRS errors occur.

39.42.1 Sender Rewriting Scheme (SRS) controls ([srs_domain](#), [srs_maxage](#), [srs_secrets](#))

39.42.1.1 [srs_domain](#) (domain-name)

(New in JES MS 6.3-1.04.) The [srs_domain](#) MTA option must be set to the domain to use in SRS addresses. Email sent to this domain must always be routed to a system capable of 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.

39.42.1.2 [srs_maxage](#) (integer)

(New in JES MS 6.3-1.04.) The [srs_maxage](#) MTA option optionally specifies the number of days before an SRS address times out. The default if the option isn't specified is 14 days.

39.42.1.3 [srs_secrets](#) (comma-separated list of strings)

(New in JES MS 6.3-1.04.) The [srs_secrets](#) MTA option takes as argument 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.

39.42.2 SRS MTA options: token_char (integer position of ASCII character)

RESTRICTED.

The token_char MTA option controls what character represents a token in the local-part of addresses. This is relevant for [SRS address handling](#). The value of this option is an integer corresponding to the ASCII character value in decimal. The default is 61, corresponding to the equal sign, =.

39.43 Syslog MTA options

The MTA has a number of options relating to generating syslog notices when certain events occurs---forms of event notices. The MTA can also optionally be configured to direct its normal [transaction logging](#) to syslog. For options relating specifically to the format of transaction logging, see [Transaction logging MTA options](#).

39.43.1 Syslog MTA options: held_sndopr (0 or 1)

The held_sndopr MTA option controls the production of syslog messages (on UNIX) when a message is forced into a held state due to certain suspicion thresholds. (Note that there are other potential causes of messages becoming .HELD, which are *not* covered by held_sndopr -- cases corresponding to explicit MTA administrator action such as execution of a `imsmta qclean` command, or cases where the held state can be logged as part of normal logging, such as execution of a [Sieve filter](#) hold action, either in an explicit Sieve filter or as a Sieve scriptlet executed due to a spam/virus filter package verdict, or application of an [address-based *_ACCESS mapping table](#) \$H flag where syslog message generation can be performed via \$< flag, or routing to the [hold channel](#) due to a [user status](#) or [domain status](#) of hold. held_sndopr is meant to warn of cases that might otherwise be easier to miss noticing, especially for unanticipated incoming problem messages.)

Suspicious cases where held_sndopr causes a syslog message include:

- A message may be forced into a held state because it has too many Received: header lines (see the various [max_*received_lines MTA options](#) for additional information):

```
HELDMSG, Header count exceeded; message has been marked .HELD automatically.
```

- Or a message may be forced into a held state because it has too many recipients (see the [holdlimit](#) channel option for more details):

```
HELDMSG, Max recipient count exceeded; message has been marked .HELD automatically.
```

- Or a message may be forced into a held state because it has too many MIME parts or levels (see the [max_mime_levels](#) and [max_mime_parts](#) MTA options):

HELDMSG, Max MIME part/level limit exceeded; message has been marked .HELD automatically.

A value of 1 for held_sndopr instructs the MTA to issue such messages when such cases occur. A value of 0 (the default) turns off these messages. The syslog messages will be issued using the configured [sndopr_priority](#) facility and severity.

39.43.2 Syslog MTA options: log_connections_syslog (integer)

The `log_connections_syslog` MTA option causes sending [MTA connection transaction log file entries](#) to syslog (UNIX). 0 is the default and means no syslog logging is performed. Setting the option to a non-zero value causes syslog logging. The absolute value sets the syslog facility/severity mask. Negative values disable the generation of the regular MTA connection transaction log file entries (which would otherwise be written to `mail.log*` or to `connection.log*`, if [separate_connection_log](#) is enabled).

Note that in JES MS 6.2 and earlier, the length of the MTA output line sent to syslog is limited to 256 characters. For JES MS 6.3 and later, the limit is 4096 characters.

The [log_messages_syslog](#) MTA option operates analogously for MTA message transaction log entries.

39.43.3 Syslog MTA options: log_messages_syslog (integer)

The `log_messages_syslog` MTA option enables sending [MTA message transaction log file entries](#) to syslog (UNIX). 0 is the default and means no syslog logging is performed; setting the option to a non-zero value causes MTA message transaction log file entries to be written to syslog. The absolute value of any non-zero value sets the syslog priority and facility mask. The relation is as follows:

$$\text{value} = \text{facility} * 8 + \text{priority}$$

The syslog priority levels and their normal meanings are:

Table 39.33 syslog priority values and their meanings

Value	Severity	syslog Keyword	Description	Action
0	Emergency	emergency	Urgent action required	System is unusable.
1	Alert	alert	Immediate action required	Should be corrected immediately, therefore notify staff who can fix the problem.
2	Critical	crit	Critical conditions	Should be corrected immediately, but indicates failure in a secondary system.
3	Error	err (error)	Error conditions	Non-urgent failures, these should be relayed to

log_messages_syslog MTA
option

				developers or admins; each item must be resolved within a given time.
4	Warning	warning (warn)	Warning conditions	Warning messages, not an error, but indication that an error will occur if action is not taken.
5	Notice	notice	Normal but significant condition	Events that are unusual but not error conditions - might be summarized in an email to developers or admins to spot potential problems - no immediate action required.
6	Informational	info	Informational messages	Normal operational messages - may be harvested for reporting, measuring throughput, etc. - no action required.
7	Debug	debug	Debug-level messages	Info useful to developers for debugging the application, not useful during operations.

The predefined syslog facility codes include:

Table 39.34 syslog facility codes and their meanings

Value	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0 (local0)
17	local use 1 (local1)

18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

For example, if informational level logging is desired under the mail facility, the value would be $2*8+6=22$.

See [RFC 5424](#) for additional information about syslog semantics and the syslog protocol.

Negative values will disable the generation of the regular `mail.log` entries; positive values mean that the syslog entries are generated in addition to the regular `mail.log*` entries.

The `log_connections_syslog` MTA option operates analogously for MTA connection transaction log entries.

Prior to Messaging Server 7.5, positive values of `log_messages_syslog` would not affect header logging (would not cause the entries due to enabling `log_header` to be copied to syslog). (The `log_header` entries, if enabled, would only be sent to syslog if the generation of regular `mail.log*` entries were disabled via negative values of `log_messages_syslog`.) As of Messaging Server 7.0.5, enabling `log_messages_syslog` normally also applies to any entries due to setting `log_header`. But setting bit 16 (value 65536) of `log_messages_syslog` will disable the sending of the header entries to syslog, thus restoring the older behavior (of not including the header lines) if desired. That is, the lowest 16 bits of `log_messages_syslog` specify the priority and facility mask as always, while bit 16 (value 65536) will, if set, suppress header logging to syslog.

Note that in JES MS 6.2 and earlier, the length of the MTA output line sent to syslog was limited to 256 characters. For JES MS 6.3 and later, the limit is 4096 characters.

Note that in JES MS 6.2, but not in JES MS 6.3 and later, it was necessary to have the `configutil` parameter `logfile.imta.syslogfacility` (corresponding to the Unified Configuration `syslogfacility` option) set (e.g., to `mail`) to have the `log_messages_syslog` MTA option take effect.

39.43.4 Syslog MTA options: log_sndopr (0 or 1)

The `log_sndopr` MTA option controls the production of syslog messages (UNIX) by the [MTA message transaction and connection logging facility](#). If this feature is enabled by specifying a value of 1, the logging facility will produce a message if it encounters any difficulty writing to its log file. A value of 0 (the default) turns off these messages. The `sndopr_priority` option controls the syslog level (facility and severity) of the syslog messages generated.

39.43.5 Syslog MTA options: sndopr_priority (0-127)

The `sndopr_priority` MTA option sets the syslog level (facility and severity) of syslog messages generated by the MTA under certain circumstances, including most cases where the MTA would generate a syslog message for an "event notice" type purpose. In particular, MTA syslog messages affected by `sndopr_priority` include:

- problems creating or deleting message files
- problems creating temporary files while buffering incoming SMTP messages
- (new in 8.0) [MESSAGE-SAVE-COPY mapping table](#) problems renaming or copying a message file
- an empty or otherwise invalid format message file is found in the MTA disk queue area
- problems updating MTA message or association counters
- problems executing the [imsimta reload utility](#)
- if [log_sndopr](#) is set, problems writing to the [MTA transaction log files](#)
- if [held_sndopr](#) is set, certain cases of sidelining of messages as `.HELD` files
- if [spamfilterN_optional](#) if set to `-2` or `2`, and trouble occurs getting a result back from the Nth spam/virus filter package
- notices configured in the [LOG_ACTION mapping table](#) or various [access mapping tables](#) that happen to support generating syslog notices via the `$<` and `$>` flags

The default is 5 (that is, LOG_NOTICE on UNIX).

Note that `sndopr_priority` does *not* affect the optional copying of MTA transaction entries to syslog, which would more typically be serving a normal logging purpose rather than an "event notice" type warning purpose; the syslog level of MTA transaction entries are instead separately controlled by the exact values of [log_messages_syslog](#) or [log_connections_syslog](#), as relevant.

39.43.6 Spamfilter MTA options: `spamfilterN_optional (-2, -1, 0, 1, 2, 3, 4)`

The `spamfilterN_optional` MTA options control the MTA's reaction when spam/virus filter package N does not respond.

By default (`spamfilter*_optional=0`), when use of a spam/virus filter package such as Brightmail is configured, a failure to initially connect to the spam/virus filter package, or a failure to get a response from the spam/virus filter package once the filter package has begun processing the envelope addresses or the message itself, will normally result in a temporary error of "4.7.1 filtering/scanning error". (The exact SMTP errors are "452 4.3.0 filtering/scanning error" if the package cannot even be contacted initially, "450 4.7.1 filtering/scanning error" if the package error occurs attempting to process the MAIL FROM: (envelope From:) argument; "452 4.7.1 filtering/scanning error" if the package error occurs attempting to process a RCPT TO: (envelope To:) argument, or "451 4.7.1 filtering/scanning error" if the package error occurs attempting to process the DATA (the message itself). Alternate text in this error message may be configured via the correspondingly numbered [error_text_spamfilterN_error](#) MTA options.) Note that for an incoming SMTP message, such a temporary error means that the message is (temporarily) rejected with that error, while for a message that is already on the system and being processed by a `reprocess/process/conversion` sort of channel, such a temporary error means that the message is reenqueued to the [reprocess channel](#) (for the `reprocess` channel to subsequently reattempt the virus/spam filter package processing).

But when `spamfilter*_optional=1` is set, the MTA's message processing will continue even if the spam/virus filter package cannot be accessed or does not complete its processing; that is, messages will be passed through without spam/virus filter package scanning (omitting spam/virus filter package scanning) if the spam/virus filter package scanning is not functioning.

New in JES MS 6.2 is support for values -2 and 2. Setting a value of 2 is similar to the effect of 1, except that a syslog notice will be generated in case of spam/virus filter package errors. A value of -1 is (currently) equivalent in effect to a value of 0. A value of -2 is similar to a value of 0, except that a syslog notice will be generated in case of spam/virus filter package errors.

New in JES MS 6.3 is support for values 3 and 4. A value of 3 tells the MTA that in case of a virus/spam filter package failure during attempted processing of an incoming message, to accept the message and queue it to the reprocess channel (for subsequent reattempted processing through the virus/spam filter package by the reprocess channel). A value of 4 does the same thing, but also logs the virus/spam filter temporary failure to syslog.

For most site's purposes, either a setting of -2 (meaning to temporarily reject the message, and generate a syslog notice logging the trouble occurrence), or (new in JES 5) a setting of 4 (meaning to defer the message to the reprocess channel, and generate a syslog notice logging the trouble occurrence) will be desirable.

39.44 Transaction logging MTA options

The MTA has a number of options affecting MTA transaction logging (and thus to some extent MTA monitoring). In particular, there are a number of MTA options that affect the format and information recorded to the [MTA message transaction log file](#) and [MTA connection transaction log file](#), and the information included in [LOG_ACTION mapping table](#) probes.

Additional MTA options relating in somewhat different ways to MTA logging may be found elsewhere:

- **TCP/IP-channel-specific options:** among these options are several specifically relating to MTA transaction logging including [MAX_B_ENTRIES](#), [MAX_J_ENTRIES](#), [LOG_BANNER](#), and [LOG_TRANSPORTINFO](#), plus a discussion of the TCP/IP channel level version of the [LOG_CONNECTION](#) option (which overrides on a per-channel basis the general, MTA-level `mta.log_connection` MTA option);
- **File format MTA options:** the [log_alq](#) and [log_deq](#) MTA options on OpenVMS affect the efficiency of MTA transaction log file handling;
- **Syslog MTA options:** these options affect syslog messages the MTA can generate as a form of *event notice*, including such a syslog message in case of a difficulty performing MTA logging ([log_sndopr](#)), certain cases of sidelining a message as `.HELD` ([held_sndopr](#)), problems with spam/virus filter package responsiveness ([spamfilterN_optional](#)), or even redirection or duplication of MTA transaction log entries to syslog ([log_messages_syslog](#) and [log_connections_syslog](#));
- **Counters MTA options:** these options affect the [MTA counters](#), which are intended for purposes of monitoring the *trend and health* of the e-mail system, rather than for precise message tracking;
- **Debug MTA options:** the [log_debug](#) MTA option to cause debugging of MTA transaction logging and channel counters update operations.

See also [MTA transaction logging](#) for additional discussion of MTA log management and transaction log format.

The (new in JES MS 6.3-0.15) XML format for MTA message transaction and MTA connection transaction logging is much more tolerant of extensions; plus it is a new format in any case. In the past, as new logging options were added, the default was that such options were disabled, so that upgrades would not unilaterally change the MTA's transaction logging format and thereby "break" existing transaction log parsers used by sites. However, with XML format which is a new format, and where parsers should be written to ignore non-understood attributes, this is not a concern. Therefore, as of JES MS 6.3-1.04, whenever XML format is enabled (`log_format=4`), many of the optional logging options default to being enabled, rather than defaulting to being disabled (as with any format other than XML format). Such options include `log_filename`, `log_filter`, `log_message_id`, `log_notary`, `log_priority`, `log_process`, `log_queue_time`, `log_reason`, and `log_username`. As of the 7.0.5 release, this also includes `log_auth`, `log_delivery_flags`, and `log_imap_flags`.

39.44.1 Transaction logging MTA options: `log_alternate_recipient` (0-3)

(New in MS 8.0.1.) The `log_deliver_by` MTA option controls whether or not any alternate recipient is included in [MTA message transaction log entries](#) and/or [LOG_ACTION mapping table](#) probes. Setting bit 0 (value 1) causes the alternate recipient value to be logged immediately after the [SMTP DELIVERBY value](#) is logged, before the [intermediate address](#). An "ab" attribute is used in the [XML log format](#). If bit 1 (value 2) is set in the `log_alternate_recipient` MTA option, then this information appears in the [LOG_ACTION](#) mapping table probe immediately after the [DELIVERBY](#), before the [intermediate address](#).

39.44.2 Transaction logging MTA options: `log_auth` (0-63)

The `log_auth` MTA option has the same basic semantics as `log_username`, except that if set `log_auth` will cause logging of the value of the SMTP MAIL FROM's AUTH parameter on enqueue, assuming one was specified and retained. (Note that this is the AUTH parameter from the MAIL FROM command, which potentially differs from whatever might be the authenticated identity specified via an SMTP AUTH command.) On dequeue the AUTH parameter is only logged if it is passed on to the remote SMTP server. The SMTP AUTH parameter field resulting from setting `log_auth` appears immediately after the username field in the old style MTA message transaction log format. An "au" attribute is used in the new XML format (`log_format=4`).

Enabling XML format transaction logging, `log_format=4`, causes the default for `log_auth` to be 1 (SMTP AUTH parameter field logged); with any other format, the default is 0 (SMTP AUTH parameter field not logged). If bit 1 (value 2) is set in the `log_auth` MTA option, then the SMTP AUTH parameter field appears in the [LOG_ACTION mapping table](#) probe immediately after the username field.

39.44.3 Transaction logging MTA options: `log_callout_delays` (0-3)

The MTA can [optionally use timers](#) to measure the total time the MTA spends waiting for various external components to return a response. The `log_callout_delays` MTA option controls the logging of this timer information. Bit 0 (value 1), if set, causes the callout logging information to be logged immediately after delivery flags in "E" (enqueue) log records. The XML attribute name in XML format logs is "cd". Bit 1 (value 1), if set, causes callout logging information to be included in the [LOG_ACTION](#) mapping probe, again immediately after delivery flags. In both cases the information is formatted as described below.

Timings are done on a per-message basis.

The following callout timers have been implemented:

- Time spent waiting for spam filters 1-8. (S1 - S8)
- Time spent waiting for [mapping routine callouts](#). (MC)
- Time spent in the following specific mappings waiting for routine callouts:
 - [PORT_ACCESS](#) (PA)
 - [FROM_ACCESS](#) (FA)
 - [ORIG_SEND_ACCESS](#) (OSA)
 - [SEND_ACCESS](#) (SA)
 - [ORIG_MAIL_ACCESS](#) (OMA)
 - [MAIL_ACCESS](#) (MA)
 - [AUTH_REWRITE](#) (AW)
 - [REVERSE](#) (RV)
 - All [Sieve mappings](#) (S)
- [Rewrite rule routine callouts](#). (RR)
- Time spent creating an SMTP transaction for the MTA to process; (note that this necessarily includes MTA processing time). (STT)
- Time spent writing the message file(s) to the MTA queue area. (QW)

When logging this information, it is formatted in the order and with delimiters as follows (note that the field names are specified in the preceding list):

```
S1 , S2 , S3 , S4 , S5 , S6 , S7 , S8 : MC , PA , FA , OSA , SA , OMA , MA , AW , RV , S : RR : STT , QW
```

Each value appears an an integer time in centiseconds followed by a semicolon and an integer use count:

```
T ; U
```

Any value consisting of a zero-time;zero-use-count pair will be omitted entirely. Use counts of 1 are also omitted. In the specific case of spam filter wait timers, an outright spam filter failure

is indicated by presence of an "F" suffix. Finally, zero elements of the comma-separated sublists may be truncated from the right.

Parsers should be aware that additional, colon-delimited elements may be added to the list as a whole, comma-separated elements may be added to sublists, and even semicolon-separated elements may be added to individual timer values.

Some samples of the timers field in XML format:

```
cd="1410,,,15:::123,25"
cd="1243,,,21:::127,33"
cd="172:1855;5,,107,248,400,500,600:::4836,14"
```

39.44.4 Transaction logging MTA options: log_connection (0-1023)

The `log_connection` MTA option controls whether or not connection information, *e.g.*, the domain name of the SMTP client sending the message, is saved in the `mail.log` file (or the `connection.log` file if `separate_connection_log` is set to 1). This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 39.35 log_connection MTA option bit values

Bit	Value	Usage
0	1	When set, connection information is included in E, D and R log records. In XML-compatible format (<code>log_format</code> is 4), this information appears in the <code>ss</code> attribute within <code>en</code> elements.
1	2	When set, connection open/close/fail records are logged by message enqueue and dequeue agents such as the SMTP and X.400 clients and servers. This bit also enables use of the <code>\$T</code> flag (for causing logging) in <code>PORT_ACCESS</code> rejection entries. In XML-compatible format (<code>log_format</code> set to 4), this information is logged in <code>co</code> elements.
2	4	When set, <code>I</code> records are logged recording ETRN events.
3	8	When set, include transport information in E, D and R log records for SMTP messages even if other connection information logging is not enabled. In XML-compatible format (<code>log_format</code> is 4), this is the <code>tr</code> attribute.
4	16	When set, <code>C</code> entries may include site-supplied text from a <code>PORT_ACCESS mapping table</code> entry; the text is added to the application information field. (Prior to Messaging Server 7.0, when SMTP server processes began unconditionally probing <code>PORT_ACCESS</code> , this bit had a meaningful side-effect of causing SMTP server processes to always query <code>PORT_ACCESS</code> regardless of channel <code>*sas1*</code> option setting; prior to Messaging Server 7.0, SMTP server processes only queried <code>PORT_ACCESS</code> when the relevant channel was marked <code>maysasl</code> , <code>maysaslserver</code> , <code>mustsas1</code> , or <code>mustsas1server</code> -- or when this bit was set.)
5	32	When set, include transport information: <code>TCP MTA-IP MTA-port remote-IP remote-port</code>

		in message log enqueue entries (E* entries): . This information appears after the SMTP delivery-status/diagnostic information. In XML-compatible format (log_format set to 4), this is the tr attribute.
6	64	When set, include application information in message log enqueue entries (E* entries). (For instance, that the SMTP protocol is in use.) This information appears after the optional (see bit 5) transport information. In XML-compatible format (log_format set to 4), this is the ap attribute.
7	128	(New in 6.2-0.01.) When set, write "U" records for SMTP AUTH attempts (successes and failures). The SASL error will be recorded in the message-id field of the connection record (immediately after the optional---see bit 6---application information); the username used in the authentication will be recorded in the username field of the connection record (which is the following field of the record); the actual SMTP response sent back to the client will be recorded in the diagnostic (final) field of the connection record.
8	256	(New in 7.0-3.01.) Include source system field in LOG_ACTION mapping table probes.
9	512	(New in 7.0-3.01.) Include both application field and transport field in LOG_ACTION mapping table probes.

Thus for instance setting log_connection to 3 will result both in additional sorts of log file entries---entries showing when an SMTP connection is opened or closed---and also additional information in regular log file entries showing the name of the system connecting (or being connected to), or the channel hostname of the enqueueing channel when the enqueueing channel is not an SMTP channel. (This is a change from PMDF V5.1 and earlier, where the value was simply 0 or 1, with 1 enabling all the then-available connection logging.) TCP/IP channels have a [channel-specific option](#) that can override this setting for particular channels.

39.44.5 Logging Conversion Tags: log_conversion_tag (bitmask)

New in 7.0.5. The log_conversion_tag MTA option causes a field recording any [conversion tags](#) to be included in MTA message transaction log file entries (both "E" and "D") and/or [LOG_ACTION mapping table](#) probes.

Bit 0 (value 1) enables including the field in MTA message transaction log file entries; bit 1 (value 2) enables including the field in LOG_ACTION mapping table probes. If enabled, this field appears after the "time in queue" field ([log_queue_time](#) option) and before the IMAP flags field ([log_imap_flags](#) option). The XML format tag for this field is "tg". The default is 0 (even in XML format logging operation).

39.44.6 Transaction logging MTA options: log_deliver_by (0-3)

(New in MS 8.0.1.) The log_deliver_by MTA option controls whether or not any SMTP DELIVERBY value (see [RFC 2852](#)) is included in [MTA message transaction log entries](#) and/or [LOG_ACTION mapping table](#) probes. Setting bit 0 (value 1) causes the DELIVERBY time value, expressed as an offset in seconds from the current time, a semicolon, the DELIVERBY mode, and (if trace information is present) the trace value, to be logged immediately after the [priority](#) is logged, before the [alternate recipient](#). A "db" attribute is used in the [XML log format](#). If bit 1 (value 2) is set in the log_deliver_by MTA option, then this information appears in

the [LOG_ACTION](#) mapping table probe immediately after the [priority](#), before the [alternate recipient](#).

39.44.7 Transaction logging MTA options: `log_diagnostics` (0-3)

(New in 7.0-3.01.) Bit 0 (value 1), if set in `log_diagnostics`, causes diagnostics information to appear in certain log entries. Bit 1 (value 2), if set, causes diagnostics information to be included in [LOG_ACTION mapping](#) probes. For instance, in the case of "B" records (bad commands received by the SMTP server), the diagnostic field will show the SMTP server error response. In the case of connection close "C" records, the diagnostic file will show the reason why the connection was closed, *e.g.*, reaching some session disconnect limit. In the case of authentication "U" entries (which note are generated as connection transaction entries, rather than message transaction entries), the result of an authentication attempt is shown in the diagnostic field. Appears just after the reason field (see the `log_reason` MTA option) and before the time-in-queue field (see the `log_queue_time` MTA option). In XML-compatible format (`log_format` set to 4), diagnostic information, if enabled, appears as the `di` attribute.

This option defaults to 1 in order to maintain compatibility with previous releases, where diagnostics information was always logged.

39.44.8 Transaction logging MTA options: `log_envelope_id` (0-3)

(New in JES MS 6.1.) The `log_envelope_id` MTA option controls whether or not the envelope ID (ENVID parameter of the MAIL FROM ESMTP command) is included in `mail.log` records and [LOG_ACTION mapping](#) probes. This information may be useful when tracking a message across multiple systems. The default is 0, meaning that the field is not logged. Setting bit 0 (value 1) causes the envelope ID to be logged, immediately after the [message filename field](#) and immediately before the [message-id field](#). In XML-compatible format (`log_format` set to 4), envelope ID logging, if enabled, appears as the `ei` attribute. New in 7.0-3.01, bit 1 (value 2), if set, causes the information to be included in [LOG_ACTION mapping](#) probes.

39.44.9 Transaction logging MTA options: `log_filename` (0-3)

The `log_filename` MTA option controls whether or not the names of the files in which messages are stored are saved in the `mail.log` file or included in [LOG_ACTION mapping](#) probes. Setting bit 0 (value 1) enables file name logging. When file name logging is enabled, the file name will appear as the first field after the final form envelope To: address. In XML-compatible format (`log_format` set to 4), file name logging, if enabled, appears as the `fi` attribute. As of JES MS 6.3p1, enabling XML format (`log_format` set to 4) causes the default for `log_filename` to be 1 (filename logging enabled); with any other format, the default is 0 (file name logging disabled), as in earlier versions. New in 7.0-3.01, bit 1 (value 2), if set, causes filename information to be included in [LOG_ACTION mapping](#) probes.

39.44.10 Transaction logging MTA options: `log_filter` (0-3)

The `log_filter` option controls whether or not any mailbox filter actions (Sieve filter actions) applicable to the message are logged in enqueue "E" records or included in [LOG_ACTION mapping](#) probes. Bit 0 (value 1), if set, causes filter information to appear in log entries. New in 7.0-3.01, bit 1 (value 2), if set, causes filter information to be included in LOG_ACTION mapping probes. This information appears after the optional "intermediate" and "original" forms of the destination address (see the [log_intermediate](#) MTA option), before the [SMTP diagnostic field](#) (which itself only appears for SMTP messages). In XML-compatible format (`log_format` set to 4), Sieve filter action(s) logging, if enabled, appears as the `f1` attribute. The filter action(s) will be enclosed within single quote characters. (In "D" records, one sees merely the two single quote characters, as no filter action is applicable during dequeue.) As of JES MS 6.3p1, enabling XML format (`log_format` set to 4) causes the default for `log_filter` to be 1 (Sieve filter action logging enabled); with any other format, the default is 0, (Sieve filter actions are not logged), as in previous versions. With `log_filter` set to 1, one might see, for instance

```
'fileinto "SPAM"'
```

or

```
'redirect "user@domain.com"'
```

As of 8.0, a "warn" clause may also be present; see the discussion in [Sieve warn extension](#).

Note that the case of a "reject" action is special, due to the inherent nature of the "reject" action. In this case, what occurs is the enqueue of a new message (a [Message Disposition Notification](#)) by the original enqueueing channel to the [process channel](#), and that new message has an implicit keep occurring. As there is no enqueue of the rejected message, the "reject" does not show up in the filter action field of any transaction log record.

As of MS 8.0, the maximum size of the `log_filter` field (the maximum length of the string recording what Sieve actions were applied) has been increased from 256 to 1024 characters.

39.44.11 Transaction logging MTA options: log_format 1, 2, 3, or 4)

The `log_format` option controls formatting options for the MTA message transaction log file, `mail.log`, (as well as the MTA connection transaction log file, `connection.log`, if [separate_connection_log=1](#) has been set so that connection entries are being recorded separately rather than in `mail.log`).

A value of 1 (the default) is the standard format. A value of 2 requests non-null formatting: empty address fields are converted to the string "<>". A value of 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. That is, `log_format` set to 3 causes length-count tagging of the following fields: The envelope-from address, the original-recipient address, the current-recipient address, the filename, the envelope-message-id, the message-id, the username, the connection information, the intermediate address(es), the Sieve filter action(s), the rejection reason text, the SMTP diagnostic, the transport information, the application information. A value of 4 (new in JES MS 6.3) requests XML-compatible format.

As of JES MS 6.3p1, note that setting `log_format` to 4 also changes the defaults for a number of other `log_*` options. In general, it enables much of the useful (but in other formats disabled by default) optional logging options, including: `log_filename`, `log_filter`, `log_message_id`, `log_notary`, `log_priority`, `log_process`, `log_queue_time`, `log_reason`, `log_username`, and (new options in 7.0.5) `log_auth`, `log_delivery_flags`, and `log_imap_flags`.

In the (new in JES MS 6.3) XML-compatible format (`log_format` set to 4), each message log entry appears as a single XML element containing multiple attributes and no subelements. Three elements are currently defined: `en` for message transaction (*e.g.*, enqueue/dequeue entries), `co` for connection transaction entries, and `he` for header entries (the optional additional records in the message transaction log file resulting from setting `log_header` to 1). Message transaction (`en`) elements can have the following attributes:

- `ts` - time stamp (always present)
- `no` - node name (present if `log_node` is 1)
- `pi` - process id (present if `log_process` is 1)
- `sc` - source channel (always present)
- `dc` - destination channel (field always present, though it will be empty for types of entries other than an enqueue "E" entry)
- `ac` - entry type or action (always present)
- `sz` - message size, reported in units of [MTA blocks](#) (field always present, though potentially empty for types of entries such as J or V entries)
- `so` - source address (always present)
- `od` - original destination address (field always present, though potentially empty for types of entries such as J entries)
- `de` - destination address (field always present, though potentially empty)
- `rf` - recipient flags (present if bit 0/value 1 of `log_notary` is set)
- `fi` - filename (present if bit 0/value 1 of `log_filename` is set)
- `ei` - envelope ID (present if bit 0/value 1 of `log_envelope_id` is set)
- `mt` - (new in 8.0) message tracking id and timeout, in the format `tracking-id:timeout` (present if bit 0/value 1 of `log_tracking` is set and the message has a tracking ID and/or timeout)
- `dd` - (new in 8.0) deferred delivery time (present if bit 0/value 1 of `log_timesis` set and the message has a deferred delivery time request)
- `ex` - (new in 8.0) expiry time (present if bit 2/value 4 of `log_times` is set and the message has an expiry time)
- `mi` - message ID (present if bit 0/value 1 of `log_message_id` is set)
- `us` - username (present if bit 0/value 1 of `log_username` is set and a username is available)

-
- `as` - (new in 8.0) Authenticated user's primary mail address, *i.e.*, normally the value of the user's `mail` attribute (present if bit 2/value 4 of `log_username` is set and a primary address is available)
 - `au` - (new in 7.0.5) SMTP AUTH parameter (present if bit 0/value 1 of `log_auth` is set and an AUTH parameter was present)
 - `ss` - source system (present if bit 0/value 1 of `log_connection` is set and source system information is available)
 - `se` - sensitivity (present if bit 0/value 1 of `log_sensitivity` is set)
 - `mp` - (new in 8.0) MT-PRIORITY (present if bit 0/value 1 of `log_mtpriority` is set)
 - `pr` - priority (present if bit 0/value 1 of `log_priority` is set)
 - `in` - intermediate address (present if bit 0/value 1 of `log_intermediate` is set and intermediate address information is available)
 - `ia` - initial (RCPT TO) address (present if bit 1/value 2 of `log_intermediate` is set and initial address information is available)
 - `ui` - (new in 8.0) recipient UID (present if bit 0/value 1 of `log_uid` is set and a recipient UID is available; in particular, will never be present in dequeue "D" records)
 - `mu` - (new in 7.0.5) IMAP UID and UIDVALIDITY for messages delivered via an [ims-ms](#) channel (present if bit 0/value 1 of `log_mailbox_uid` is set and UID and/or UIDVALIDITY data is available)
 - `fr` - SMTP extension FUTURERELEASE value (present if bit 0/value 1 of `log_futurerelease` is set and future release value is present)
 - `fl` - Sieve filter actions applied (present if bit 0/value 1 of `log_filter` is set and Sieve filter information is available; in particular, will never be present in dequeue "D" records since there is no Sieve filtering performed (hence no filter information) at that point)
 - `re` - reason (present if bit 0/value 1 of `log_reason` is set and a reason string is available)
 - `di` - diagnostic (present if bit 0/value 1 of `log_diagnostics` is set (the default) and diagnostic information is available)
 - `tr` - transport information (present if bit 5/value 32 of `log_connection` is set and transport information is available)
 - `ap` - application information (present if bit 6/value 64 of `log_connection` is set and application information is available)
 - `qt` - time in queue (present if bit 0/value 1 of `log_queue_time` is set)
 - `tg` - (new in 7.0.5) [conversion tags](#) (present if bit 0/value 1 of `log_conversion_tag` is set and any conversion tags are present)
 - `if` - (new in 7.0.5) IMAP flags (present if bit 0/value 1 of `log_imap_flags` is set and any [IMAP flags are present](#))
 - `df` - (new in 7.0.5) delivery flags (present if bit 0/value 1 of `log_delivery_flags` is set)

- `cd` - (new in 8.0) time spent waiting on external service callouts (present if bit 0/value 1 of `log_callout_delays` is set)
- `t1` - (new in 8.0) String produced by "transactionlog" Sieve actions

Here is a sample `en` entry (wrapped for printing clarity; the actual log file entries always appear in reality on a single line):

```
<en ts="2004-12-08T00:40:26.70" pi="0d3730.10.43" sc="tcp_local"
dc="1" 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/ZZ01LI4XPX0DTM00IKA8.00" ei="01LI4XPQR2EU00IKA8@mauve.sun.com"
mi="&lt;11a3b401c4dd01$7c1clee0$1906fad0@elara&gt;" 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"/>
```

Connection transaction (`co`) entries can have the following attributes:

- `ts` - time stamp (always present; also appears in `en` entries)
- `no` - node name (present if `log_node` is 1; also appears in `en` entries)
- `pi` - process id (present if `log_process` is 1; also appears in `en` entries)
- `sc` - source channel (always present; also appears in `en` entries)
- `dr` - direction (always present; specific to connection entries): a "+" for inbound connections, or a "-" for outbound connections
- `ac` - entry type or action (always present; also appears in `en` entries)
- `tr` - transport information (always present; may also appear in `en` entries when bit 5 of `log_connection` is set);
- `ap` - application information (always present; may also appear in `en` entries when bit 6 of `log_connection` is set)
- `mi` - the name presented on the ETRN command line (present only if bit 0/value 1 of `log_message_id` is set, and ETRN was used with an argument; note that setting bit 0/value 1 of `log_message_id` also causes the logging of message ID information, if available, in message transaction (`en`) log records)
- `us` - username (present only if bit 0/value 1 of `log_username` is set and username information is available; also appears in `en` entries)
- `ex` - (new in 8.0) in "U" records, additional authentication information such as the authentication mechanism (present only if bit 4/value 16 of `log_username` is set and such extra information is available)
- `di` - diagnostic (present only if bit 0/value 1 of `log_diagnostic` is set and diagnostic information is available; also appears in `en` entries)
- `ct` - time to connect/fail to connect (present only if bit 0/value 1 of `log_queue_time` is set; specific to connection entries)

Here is a sample `co` entry (wrapped purely for printing clarity; in reality, such entries always appear on a single line):

```
<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` is 1; also used in `en` entries)
- `pi` - process id (present if `log_process` is 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"/>
```

Note that the maximum length of MTA transaction record, both for message transaction records and connection transaction records, is 4096 characters.

39.44.12 Transaction logging MTA options: log_futurerelease (0-3)

(New in 8.0.) The `log_futurerelease` MTA option controls whether or not any `FUTURERELEASE` value is included in [MTA message transaction log entries](#) and/or [LOG_ACTION](#) mapping table probes. Setting bit 0 (value 1) causes the future release value, expressed as an offset in seconds from the current time, to be logged immediately after the [mailbox UID](#) is logged, before the [Sieve filter information](#). A `fr` attribute is used in the [XML log format](#). If bit 1 (value 2) is set in the `log_futurerelease` MTA option, then this information appears in the [LOG_ACTION](#) mapping table probe immediately after the [mailbox UID](#), before the [Sieve filter information](#).

39.44.13 Transaction logging MTA options: log_header (0, 1, 2)

The `log_header` MTA option controls whether the MTA writes message headers to the message transaction log file, `mail.log`. Or, as of JES MS 6.0, see also the channel option [logheader](#), which may be used to enable this facility on a per-channel basis, as well as to override `log_header` on a per-channel basis.

Originally, the permitted values for `log_header` were merely 0 or 1. As of at least JES MS 6.1, the option takes a bit-encoded integer value instead. A value of 1 enables message header logging for both enqueue and dequeue directions; a value of 2 enables message header logging for message enqueues, without enabling logging for message dequeues. A value of 0, the default, disables message header logging.

Note that only outermost message headers are available for logging purposes; the "inner" headers on inner MIME parts, if any, are not available. The specific headers written to the log

file are controlled by a site-supplied `log_header.opt` file. The format of this file is that of other MTA header option files. For instance, a `log_header.opt` file containing

```
To: MAXIMUM=1
From: MAXIMUM=1
Defaults: MAXIMUM=-1
```

would result in writing the first To: and the first From: header per message to the log file.

When header field logging is enabled and `log_format` is 1, 2, or 3, the header fields are logged in the format:

```
dd-mmm-yyyy hh:mm:ss.ss > header-line
```

or up to two additional fields may be present (if `log_node` is 1 and `log_process` is 1 -- the header line logging makes no display of any other fields):

```
dd-mmm-yyyy hh:mm:ss.ss node process-field > header-line
```

with one such line per header line to be logged. In XML-compatible format (`log_format` set to 4), the header fields are the elements.

39.44.14 Logging IMAP flags: `log_imap_flags` (bitmask)

New in 7.0.5. The `log_imap_flags` MTA option causes a field recording [IMAP flags \(those set by the MTA\)](#) to be included in [MTA message transaction log file entries](#) and/or [LOG_ACTION mapping table](#) probes.

Bit 0 (value 1) enables including the field in MTA message transaction log file entries; bit 1 (value 2) enables including the field in `LOG_ACTION` mapping table probes. If enabled, this field appears after the conversion tag field (`log_conversion_tag` option). The XML format tag for this field is "if". The default in non-XML format logging is 0; the default when XML format (`log_format=4`) is enabled is 1.

39.44.15 Transaction logging MTA options: `log_delivery_flags` (bitmask)

New in 7.0.5. The `log_delivery_flags` MTA option causes a field recording delivery flag bits (recorded as an integer) in [MTA message transaction log file entries](#) and/or [LOG_ACTION mapping table](#) probes. Bit 0 (value 1) enables including the field in MTA message transaction log file entries; bit 1 (value 2) enables including the field in `LOG_ACTION` mapping table probes.

If enabled, this field appears after the IMAP flags field (`log_imap_flags` MTA option). The XML format tag for this field is "df". The default in non-XML format logging is 0; the default when XML format (`log_format=4`) is enabled is 1.

39.44.16 Transaction logging MTA options: `log_intermediate` (0-15)

(New in JES MS 6.2.) The `log_intermediate` MTA option controls the inclusion of the "intermediate" form of the destination address, and the inclusion of the "original" (RCPT TO) form of the destination address, in [MTA message transaction log \(mail.log\) records](#). The option takes a bit-encoded integer value.

Table 39.36 log_intermediate MTA option bit values

Bit	Value	Usage
0	1	When set, the "intermediate" address form is logged, after the optional (see the log_priority MTA option) priority field. In XML-compatible format (log_format set to 4), the intermediate address is the <code>in</code> attribute.
1	2	When set, the "original" (RCPT TO) address form is logged. This information appears after the optional (see bit 0) "intermediate" address form and before the optional (see the log_filter MTA option) Sieve filter field. In XML-compatible format (log_format set to 4), the initial address is the <code>ia</code> attribute.
2	4	When set, the "intermediate" address form is included in LOG_ACTION mapping probes.
3	8	When set, the "original" (RCPT TO) address form is included in LOG_ACTION mapping probes.

39.44.17 Transaction logging MTA options: log_local (0 or 1)

The `log_local` MTA option controls whether or not the domain name for the local host is appended to [logged addresses](#) that don't already contain a domain name. A value of 1 enables this feature, which is useful when logs from multiple systems are concatenated and processed. A value of 0, the default, disables this feature.

39.44.18 Transaction logging MTA options: log_mailbox_uid (0-3)

Messages delivered to an IMAP store are tagged with a UID and the folder's UIDVALIDITY value upon insertion. The `log_mailbox_uid` MTA option provides the means to log this information. At present the field consists of the two values delimited by a colon. This can be useful when there is a need to correlate a message in the store with MTA actions.

The `log_mailbox_uid` MTA option defaults to 0. Setting bit 0 (value 1) logs the UID and UIDVALIDITY of messages delivered by the [ims-ms channel](#) to the store. The UID and UIDVALIDITY appears immediately after the LDAP uid. A "mu" attribute is used in the XML log format ([log_format=4](#)). If bit 1 (value 2) is set in the `log_uid` MTA option, then the uid appears in the [LOG_ACTION mapping table](#) probe immediately after the LDAP uid.

Compatibility note: Additional information will be added to this field in the future. When that happens it will appear as additional colon-separated values. Any code written to process this field needs to take this into account.

39.44.19 Transaction logging MTA options: log_message_id (0-3)

The `log_message_id` MTA option primarily controls whether or not message IDs are saved in the `mail.log` file or included in `LOG_ACTION` mapping probes. Bit 0 (value 1), if set, enables message ID logging in MTA transaction logging. When message ID logging is enabled, the message ID will be logged after the final form envelope To address entry--- and after the message file name, if `log_filename` is also enabled. In XML-compatible format (`log_format` set to 4), message ID logging, if enabled, appears as the `mi` attribute. As of JES MS 6.3p1, enabling XML format (`log_format` set to 4) causes the default for `log_message_id` to be 1 (message ID logging enabled); with any other format, the default is 0 (message ID logging disabled), as in previous versions.

Note that the `log_message_id` option's transaction log field is also used in authentication "U" records to record the SASL error, and in SMTP ETRN "I" records to record the host name from the client ETRN command.

New in 7.0-3.01, bit 1 (value 2), if set, causes message ID information to be included in `LOG_ACTION` mapping probes.

39.44.20 Logging message transfer priorities: `log_mtpriority` (bitmask)

New in 8.0. The Message Transfer Priority SMTP extension MT-PRIORITY, defined in [RFC 6710](#), provides the means to associate a message transfer priority with a given message. Logging of these priority values, ranging from -9 to 9, with a default value of 0, is controlled by the `log_mtpriority` MTA option.

Note that there is a Priority: header-based priority mechanism and associated logging option `log_priority`. These are separate mechanisms. An explicit MT-PRIORITY overrides the older Priority: header handling.

The `log_mtpriority` option defaults to 0. Setting bit 0 (value 1) enables logging of the message transfer priority associated with each transaction. The message transfer priority appears immediately after the message sensitivity and before the header-based priority in each log entry. A "mp" element is used in the new XML log format (`log_format=4`). If bit 1 (value 2) is set in the `log_mtpriority` MTA option, then the message transfer priority appears in the `LOG_ACTION` mapping table probe, again immediately after the sensitivity field and before the header-based priority field.

39.44.21 Transaction logging MTA options: `log_node` (0 or 1; OpenVMS only until JES MS 6.3)

The `log_node` MTA option controls whether or not the node associated with the process generating an entry is saved in the `mail.log` file (and `connection.log` file, if a separate connection log file is used). This may be useful information when the MTA is running in a multi-node cluster.

A value of 1 enables node name logging. When the node name is logged, it will appear as the first field following the date and time stamps in log entries. In XML-compatible format (`log_format` set to 4), node logging, if enabled, appears as the `no` attribute. A value of 0 (the default) disables node name logging. (Note that on UNIX and Windows, prior to JES MS 6.3, the node name was always considered to be null, so on UNIX and Windows, attempting to set `log_node` to 1 merely resulted in an empty node field; *i.e.*, an extra space after the time stamp in formats other than XML format. As of JES MS 6.3, if `log_node` is 1, then the MTA

on UNIX will log the result of a `gethostname` call when that call found a value, or the "L" channel official host name, otherwise.)

39.44.22 Transaction logging MTA options: `log_notary` (0-3)

The `log_notary` MTA option controls whether the MTA includes an indicator of NOTARY (delivery receipt) flags in the `mail.log` file entries or in `LOG_ACTION` mapping probes. Bit 0/value 1, if set, enables NOTARY flag logging. As of JS MS 6.3P1, enabling XML format (`log_format` set to 4) causes the default for `log_notary` to be 1; with any other format, the default is 0, as in previous versions. The NOTARY flags will be logged as a bit encoded integer after the current form of the envelope To: address. In XML-compatible format (`log_format` set to 4), notification flag logging, if enabled, appears in the `rf` attribute.

Table 39.37 `log_notary` MTA option bit value

Bit	Value	Usage
0	1	Use header-style delivery receipt requests, corresponding to the use of the <code>reporthead</code> channel option or <code>reportboth</code> channel option.
1	2	Suppress header-style delivery receipt requests.
2	4	NDN failure notification requested: Generate an NDN if the message fails delivery, <i>i.e.</i> , NOTARY parameter NOTIFY=FAILURE.
3	8	DSN delivery receipt requested: Send a DSN (delivery receipt) if message delivery succeeds, <i>i.e.</i> , NOTARY parameter NOTIFY=SUCCESS.
4	16	DSN delay warning requested: Send a "delivery of your message has been delayed" DSN if message delivery is delayed, <i>i.e.</i> , NOTARY parameter NOTIFY=DELAY.
5	32	Internal flag keeping track of notification handling for mailing lists.
6	64	Suppress all notifications, <i>i.e.</i> , NOTARY parameter NOTIFY=NEVER.
7	128	Generate a delay DSN if the first delivery attempt fails.
8	256	NOTARY was used; set whenever any NOTIFY parameter was used during envelope From submission.

New in 7.0-3.01, bit 1 (value 2), if set, causes NOTARY information to be included in `LOG_ACTION` mapping probes.

39.44.23 Transaction logging MTA options: `log_priority` (0-3)

The `log_priority` MTA option controls whether or not the effective processing priority for the message is included in the `mail.log` file entries or `LOG_ACTION` mapping probes. Bit 0/value 1, if set, enables priority logging. As of JES MS 6.3P1, enabling XML format (`log_format` set to 4) causes the default for `log_priority` to be 1; with any other format, the default is 0, as in previous versions. If logging is enabled, the priority will be logged after the message sensitivity value, before the transport information. In XML-compatible format (`log_format` set to 4), priority logging, if enabled, appears as the `pr` attribute.

New in 7.0-3.01, bit 1 (value 2), if set, causes priority information to be included in `LOG_ACTION` mapping probes.

As of 8.0, the MTA also supports the SMTP MT-PRIORITY extension, which provides a different, non-header-based, priority facility. The logging of MT-PRIORITY is controlled separately, via the [log_mtpriority](#) MTA option.

39.44.24 Transaction logging MTA options: log_process (0 or 1)

The `log_process` MTA option controls whether or not the ID of the process is saved in the `mail.log` file (and the `connection.log` file, if a separate connection log is used).

A value of 1 enables process ID logging. A value of 0 disables it. As of JES MS 6.3p1, enabling XML format (`log_format` set to 4) causes the default for `log_process` to be 1; with any other format, the default is 0, as in previous versions.

The process ID will be logged after the date and time stamps in log entries---and after the node name, if `log_node` is also enabled. In XML-compatible format (`log_format` set to 4), process ID logging, if enabled, appears as the `pi` attribute. The process ID field itself will consist of the process ID in a hexadecimal representation followed by a period, next in the case of a multithreaded channel the thread ID followed by a period, followed by a (per process) counter for logging operations. That is, in the case of a single threaded channel

process-id.counter

or in the case of a multithreaded channel

process-id.thread-id.counter

Note in particular that via the process id and thread id, TCP/IP channel message enqueue/dequeue (E/D) records may be correlated with SMTP connection open/close (O/C) records. `log_process` is one of the most useful MTA logging options for correlating between multiple transaction records relating to the same message, observing the timing between entries, *etc.*

39.44.25 Transaction logging MTA options: log_queue_time (0-3)

(New in JES MS 6.3.) The `log_queue_time` MTA option controls whether "time in queue" information, measured in seconds, is included in [MTA message transaction log records](#) or [LOG_ACTION mapping](#) probes.

Bit 0/value 1, if set, enables such logging; having the bit clear disables such logging. As of JES MS 6.3P1, enabling XML format (`log_format` set to 4) causes the default for `log_queue_time` to be 1; with any other format, the default is 0, as in previous versions.

If logging is enabled, then the time that the message has been in this channel queue (the current system time minus the creation time at which this message file was generated as stored in the envelope information in the message file) will be logged after the optional `log_connection` bit 6 (value 64) application information. (In versions prior to 7.0.5, this made queue time the very last field logged. But some optional new fields added in 7.0.5 can appear after queue time.) Note that the final ordering is: `log_filter` field, `log_reason` field, `log_diagnostics` diagnostic field, transport field (`log_connection` bit 5/value 32), application field (`log_connection` bit 6/value 64), queue time field. (Note that queue time is not a counted-length field; `log_format` set to 3 has no effect upon this field.) When

log_format is set to 4 so that XML format is output, queue time logging, if enabled, appears in the qt attribute.

New in 7.0, the queue time field also contains meaningful (though slightly different in meaning) information on LMTP server back end hosts: there it represents how long the message transaction took (the time from start of the message transaction until message deposit into the store and acknowledgement back to the LMTP client). Also new in 7.0, setting log_queue_time to 1 will cause inclusion of a final field in MTA connection log O and Y records showing the time it took to open the connection, or fail to open a connection, respectively. (This is the time that it took from just before the MTA begins attempting any appropriate DNS lookups, until writing its connection log entry.)

New in 7.0-3.01, bit 1 (value 2), if set, causes queue time information to be included in LOG_ACTION mapping probes.

39.44.26 Transaction logging MTA options: log_reason (0-3)

(New in JES MS 6.3.) The log_reason MTA option controls whether message rejection reason text is included in MTA message transaction log entries and LOG_ACTION mapping probes that correspond to a message rejection ("R" or "K" records).

Setting bit 0/value 1 enables such logging; clearing the bit disables such logging. As of JES MS 6.3P1, enabling XML format (log_format set to 4) causes the default for log_reason to be 1; with any other format, the default is 0, as in previous versions.

If logging is enabled, the reason text string will be logged just after the filter field (log_filter set to 1) and just before the SMTP delivery status (log_diagnostics SMTP diagnostic) field. In XML-compatible format (log_format set to 4), the reason logging, if enabled, appears as the re attribute.

New in 7.0-3.01, bit 1 (value 2), if set, causes reason information to be included in LOG_ACTION mapping probes.

39.44.27 Transaction logging MTA options: log_sensitivity (0-3)

The log_sensitivity MTA option controls whether message Sensitivity: header values are included in MTA message transaction log entries or LOG_ACTION mapping probes.

Bit 0/value 1, if set, enables such logging; clearing the bit disables such logging. If logging is enabled, the sensitivity value will be logged in an integer representation after the connection information, before the priority field. In XML-compatible format (log_format set to 4), sensitivity logging, if enabled, appears in the se attribute. The values are (with "Normal" being the default, intended for less sensitive material, and "Company-Confidential" corresponding to the most sensitive material):

Table 39.38 log_sensitivity MTA option values

Value	Meaning
0	Normal
1	Personal

2	Private
3	Company-Confidential

New in 7.0-3.01, bit 1 (value 2), if set, causes sensitivity information to be included in [LOG_ACTION mapping](#) probes.

39.44.28 Transaction logging MTA options: log_times (0-15)

The `log_times` MTA option controls whether deferred delivery time request values -- and optionally message expiry time values -- are included in [MTA message transaction log entries](#) or [LOG_ACTION mapping](#) probes.

Bit 0/value 1, if set, enables logging any deferred delivery request time, as for instance requested via the [FUTURERELEASE SMTP extension](#) or [Deferred-delivery: header line](#). Clearing bit 0 disables such logging. In XML-compatible format (`log_format` set to 4), the deferred delivery time logging, if enabled, appears in the `dd` attribute. Bit 1/value 2, if set, causes a deferred delivery request time field to be included in [LOG_ACTION mapping](#) probes.

Bit 2/value 4, if set, enables logging any message expiry time. In XML-compatible format (`log_format` set to 4), the message expiry logging, if enabled, appears in the `ex` attribute. Setting bit 3/value 8 causes a message expiry field to be included in [LOG_ACTION mapping](#) probes.

If enabled, the deferred delivery time and message expiry time fields appear immediately after the [tracking ID](#) field and immediately prior to the [message ID](#) field.

39.44.29 Log Tracking Information (log_tracking)

The `log_tracking` MTA option controls logging of message tracking/recall information. Bit 0 (value 0), if set, includes the tracking id and the current tracking in transaction log entries. The two appear as a single field; the id appears first and is separated from the timeout value by a colon. Tracking information appears immediately after the [envelope id](#) and before the [deferred delivery time](#). An "mt" attribute is used in the XML format (`log_format=4`).

If bit 1 (value 2) is set in the `log_tracking` MTA option, then the tracking id and timeout, again separated by a colon, appear in the [LOG_ACTION mapping table](#) probe, again immediately after the [envelope id](#) and before the [deferred delivery time](#).

39.44.30 Transaction logging MTA options: log_transactionlog (0-3)

(New in 8.0.) The `log_transactionlog` MTA option controls whether [Sieve transactionlog action](#) strings are included in [MTA message transaction log records](#). The option defaults to 0, meaning that such Sieve actions are not logged. Setting bit 0 (value 1) causes the `transactionlog` string to be logged at the very end of enqueue ("E") records. The XML attribute name in XML format logs (`log_format=4`) is "t1". Setting bit 1 (value 2) causes the the `transactionlog` string to be included in the [LOG_ACTION mapping table](#) probe, again at the very end.

For instance, with MTA message transaction logging enabled (the `logging channel` option set for all channels) and with `log_transactionlog` also set:

```
msconfig> show logging
role.channel:defaults.logging
msconfig> set log_transactionlog 1
```

then an MTA system filter (see for instance the `msconfig` command `edit filter`) including:

```
require "variables";
if header :matches "subject" "*" {transactionlog "${0}";}
```

will cause MTA message transaction log records to include the contents of the Subject: header line. (Note that merely logging the Subject: header line of messages passing through the MTA could instead be achieved via use of `log_header` or `logheader`. But the Sieve script approach allows more fine-tuning, as a Sieve script can be coded with complex logic dependent upon other message details, such as message sender or recipient, presence of specific strings in the header, *etc.*)

39.44.31 Transaction logging MTA options: log_uid (0-3)

Certain alias operations, particularly alias expansion of user addresses, involve looking up LDAP entries with uid attributes. When such entries are encountered, the uid is carried through the uid expansion process and, in the case of delivering to the Message Store, the uid is typically incorporated into the resulting address. The `log_uid` MTA option provides the means to log such uids. This can be useful when there is a need to identify the last LDAP entry involved in the alias expansion. Note that uids are only logged on message enqueue operations; there is no uid available to log on message dequeues.

The `log_uid` MTA option defaults to 0. Setting bit 0 (value 1) logs any available uid. The uid appears immediately after the initial recipient address. A "ui" attribute is used in the XML log format (`log_format=4`). If bit 1 (value 2) is set in the `log_uid` MTA option, then the uid appears in the [LOG_ACTION mapping table](#) probe immediately after the initial destination address field.

39.44.32 Transaction logging MTA options: log_use_xtext (bitmask)

The `log_use_xtext` MTA option controls the use of xtext encoding of addresses in [MTA transaction log file](#) entries and [LOG_ACTION mapping](#) probes.

39.44.33 Transaction logging MTA options: log_username (0-63)

The `log_username` MTA option controls whether or not the username associated with a process that enqueues mail is saved in the [mail.log file \(and optional connection.log file\)](#) or included in [LOG_ACTION mapping](#) probes.

Note that messages submitted via SMTP with authentication (SMTP AUTH) will be considered to be owned by the username that authenticated, prefixed with the asterisk, *, character. (Note

also that messages enqueued to the [filter_discard channel](#) always get the enqueueing username set to literally FILTER_DISCARD; this is for access control purposes.)

Bit 0/value 1, if set, enables username logging. When username logging is enabled, in the MTA message transaction log entries the username will be logged after the final form envelope To address field in log entries---and after the message ID, if [log_message_id](#) is enabled also, before the connection information, if bit 0/value 1 of [log_connection](#) is enabled also. In the MTA connection transaction log entries, the username will be logged after the application information (and after the optional field present if [log_message_id](#) is enabled). In XML-compatible format ([log_format](#) set to 4), the username logging, if enabled, appears in the `us` attribute. A value of 0 disables username logging. As of JES MS 6.3p1, enabling XML format ([log_format](#) set to 4) causes the default for `log_username` to be 1; with any other format, the default is 0, as in previous versions.

New in 7.0-3.01, bit 1 (value 2), if set, causes username information to be included in [LOG_ACTION mapping](#) probes.

New in 8.0, bit 2 (value 4) enables inclusion of the primary mail address associated with the authenticated username to be logged, XML tag "as"; bit 3 (value 8) causes that same address to be included in [LOG_ACTION mapping table](#) probes. This address appears immediately after the username in both the transaction log, and in the mapping table probe. Also new in 8.0, bit 4 (value 16) causes additional information associated with an authentication operation (currently merely the name of the authentication mechanism), XML tag "ex"; additional information may be added in the future) to be included in "U" records; bit 5 (value 32) causes the same information to be included in [LOG_ACTION mapping table](#) probes immediately after the authenticated username.

As of JES MS 6.3p1, enabling XML format ([log_format=4](#)) causes the default for `log_username` to be 1; with any other format, the default is 0, as was always the case in previous versions.

39.44.34 Transaction logging MTA options: **separate_connection_log (0 or 1)**

The `separate_connection_log` MTA option controls whether the connection log information generated by setting [log_connection](#) to 1 is stored in the usual [MTA message transaction logging files, mail.log*, or stored separately in connection.log* files](#). A `separate_connection_log` value of 0, the default, causes connection transaction logging to be stored in the regular message transaction log files; a value of 1 causes the connection transaction logging to be stored separately.

39.44.35 Transaction logging MTA options: **return_split_period (integer)**

The `return_split_period` MTA option affects the frequency of MTA transaction log file roll over. The MTA will start new [mail.log_current and connection.log_current files](#) every `return_split_period` runs of the return job. That is, if `return_split_period` has the value N, then new log files will be started on every Nth run of the return job.

For normal operation, a value of 1 is recommended for this option. That causes new log files to be started every time the return job is run. If, however, the return job is being

run hourly via the `schedule.task:return_job.crontab` option, then changing `return_split_period` to the value 24 is recommended. That causes new log files to only be started once per day despite the return job running once per hour.

39.44.36 Transaction logging MTA options: `return_cleanup_period` (integer)

Sites may supply a cleanup script, `SERVERROOT/data/site-programs/bin/daily_cleanup`, which will then be run every time the return job runs. Such a script might, for instance, move or rename [MTA transaction log files](#). To only run this site-supplied script on every Nth run of the return job, set the value of the `return_cleanup_period` MTA option to N.

39.45 OpenVMS user agent MTA options

A number of long-standing MTA options exist that, relevant only on OpenVMS, affect the MTA's interaction with OpenVMS user agents such as VMS MAIL, PMDF MAIL, and DECwindows MAIL. These MTA options are not relevant on other platforms.

39.45.1 `delivery_receipt_off` Option

OpenVMS only.

This option is used to specify a special [RFC 822](#) comment string used in IN% addresses in VMS MAIL to disable any requests for a delivery receipt. The default if this option is not specified is:

```
(NO-DELIVERY-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

39.45.2 `delivery_receipt_on` Option

OpenVMS only.

This option is used to specify a special [RFC 822](#) comment string used in IN% addresses in VMS MAIL to request a delivery receipt. The default if this option is not specified is:

```
(DELIVERY-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

39.45.3 OpenVMS user agent MTA options: `dis_nesting` (non-negative integer)

OpenVMS only.

The `dis_nesting` MTA option controls how many nesting levels the MTA allows in the expansion of VMS MAIL distribution lists. A value of 0 disables the ability to use VMS MAIL

distribution lists. Currently this option only affects the PMDF MAIL utility. The default value for this option is 20.

39.45.4 OpenVMS user agent MTA options: form_names (string; OpenVMS only)

OpenVMS only.

The `form_names` option specifies the names of pop-up form images. Multiple values should be separated with commas but *not* with spaces. The default is "FAX-FORM, PH-FORM, X500-FORM".

39.45.5 mail_delivery_filename Option

OpenVMS only.

39.45.6 missing_address Option

OpenVMS only.

39.45.7 multinet_mm_exclusive Option

OpenVMS only.

39.45.8 OpenVMS user agent MTA options: read_receipt_off (string)

OpenVMS only.

The `read_receipt_off` MTA option is used to specify a special [RFC 822](#) comment string used in IN% addresses in VMS MAIL to disable any requests for a read receipt. The default if this option is not specified is:

```
(NO-READ-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

39.45.9 OpenVMS user agent MTA options: read_receipt_on (string)

OpenVMS only.

The `read_receipt_on` MTA option is used to specify a special [RFC 822](#) comment string used in IN% addresses in VMS MAIL to request a read receipt. The default if this option is not specified is:

```
(READ-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

39.45.10 OpenVMS user agent MTA options: safe_tcl_mode (bitmask)

OpenVMS only.

The `safe_tcl_mode` MTA option takes a bitmask argument. The lowest bit, when set, allows components of PMDF to interpret message parts of type application/safe-tcl upon user confirmation. (At present, PMDF MAIL is the only component of PMDF which supports Safe-Tcl.) The second lowest bit, when set, puts PMDF's Safe-Tcl interpreter into a very paranoid mode in which it will not allow information from sensitive message header lines to be disclosed to Safe-Tcl scripts.

The default value for this option is 3 which allows Safe-Tcl scripts to be executed upon user confirmation, but does so in "paranoid" mode.

39.45.11 use_mail_delivery Option

OpenVMS only.

39.45.12 vms_mail_exclusive Option

OpenVMS only.



Chapter 40 MTA Tailor options

40.1 Directory location MTA Tailor options	40-2
40.1.1 imta_root MTA Tailor option	40-2
40.1.2 imta_lib MTA Tailor option	40-3
40.1.3 imta_bin MTA Tailor option	40-3
40.1.4 imta_table MTA Tailor option	40-3
40.1.5 imta_dl MTA Tailor option	40-3
40.1.6 imta_log MTA Tailor option	40-3
40.1.7 imta_tmp Option	40-3
40.1.8 imta_db_tmp Option	40-3
40.1.9 imta_queue Option	40-4
40.1.10 imta_help Option	40-4
40.1.11 imta_lang Option	40-4
40.1.12 imta_program MTA Tailor option	40-4
40.2 File name MTA Tailor options	40-4
40.2.1 imta_option_file MTA Tailor option	40-5
40.2.2 imta_system_filter_file MTA Tailor option	40-5
40.2.3 imta_config_file MTA Tailor option	40-5
40.2.4 imta_xml_config_file MTA Tailor option	40-5
40.2.5 imta_command_data MTA Tailor option	40-5
40.2.6 imta_config_data MTA Tailor option	40-6
40.2.7 imta_charset_data MTA Tailor option	40-6
40.2.8 imta_alias_file MTA Tailor option	40-6
40.2.9 imta_primary_log MTA Tailor option	40-6
40.2.10 imta_secondary_log MTA Tailor option	40-6
40.2.11 imta_tertiary_log MTA Tailor option	40-7
40.2.12 imta_primary_connection_log Option	40-7
40.2.13 imta_secondary_connection_log Option	40-7
40.2.14 imta_tertiary_connection_log Option	40-7
40.2.15 imta_name_content_file Option	40-7
40.2.16 imta_forward_data Option	40-7
40.2.17 imta_reverse_data Option	40-7
40.2.18 imta_general_data Option	40-7
40.2.19 imta_alias_database Option	40-7
40.2.20 imta_reverse_database Option	40-8
40.2.21 imta_forward_database Option	40-8
40.2.22 imta_general_database Option	40-8
40.2.23 imta_domain_database Option	40-8
40.2.24 imta_ssr_database Option	40-8
40.2.25 imta_dn_to_id_database Option	40-8
40.2.26 imta_user_profile_database Option	40-8
40.2.27 imta_charset_option_file Option	40-9
40.2.28 imta_mapping_file Option	40-9
40.2.29 imta_conversion_file Option	40-9
40.2.30 imta_hold Option	40-9
40.2.31 imta_jbc_config_file Option	40-9
40.2.32 imta_dispatcher_config Option	40-9
40.2.33 imta_libutil Option	40-9
40.2.34 imta_libmap Option	40-9
40.2.35 imta_security_config_file Option	40-10
40.3 User MTA Tailor options	40-10

40.3.1 imta_user MTA Tailor option	40-10
40.3.2 imta_user_username MTA Tailor option	40-10
40.3.3 imta_world_group MTA Tailor option	40-10
40.4 Scheduling MTA Tailor options	40-10
40.4.1 imta_version_limit Option	40-11
40.4.2 imta_version_limit_period Option	40-11
40.4.3 imta_return_synch_period Option	40-11
40.4.4 imta_return_split_period Option	40-11
40.4.5 imta_return_cleanup_period Option	40-11
40.4.6 imta_verify_return Option	40-11
40.4.7 imta_synch_cache_period Option	40-11

In legacy configuration, various MTA installation and operational parameters would be set in the MTA Tailor file of option settings. But with Unified Configuration, the MTA Tailor file is obsolete and no longer used.

Old MTA Tailor options that specified [locations of MTA directories](#) or [names of MTA configuration files](#) have typically been replaced by rationalized, consistent locations based off the installation main location and located via the SERVERROOT environment variable.

The SERVERROOT value in turn is used to construct the DATAROOT value (SERVERROOT/data) and CONFIGROOT value (SERVERROOT/config).

New in 8.0, the [recipe language](#) has a "get_path" function that takes as argument "server", "data", or "config", and returns a file path to the current server instance; *e.g.*:

```
msconfig> exec get_path "server"  
> "/opt/sun/comms/messaging64"  
msconfig> exec get_path "config"  
> "/opt/sun/comms/messaging64/config"  
msconfig> exec get_path "data"  
> "/opt/sun/comms/messaging64/data"
```

As of Messaging Server 7.0.5, [MTA Tailor options that determined the username and group of the MTA user and unprivileged user](#) have been moved to the file `restricted.cnf`.

[MTA Tailor options controlling some aspects of scheduling of MTA tasks](#) have either been incorporated into [values in Scheduler tasks](#), or replaced by various `mta.option-name` Unified Configuration options.

40.1 Directory location MTA Tailor options

As of Messaging Server 7.0-0.04, many formerly configurable (via [MTA Tailor option](#)) MTA directory locations are now hard-coded. `tmpdir` (formerly called `imta_tmp`) and `langdir` (formerly called `imta_lang`) are still configurable.

40.1.1 MTA Tailor options: `imta_root` (directory path)

DELETED.

Root of MTA installation is now determined via the SERVERROOT environment variable.

40.1.2 MTA Tailor options: `imta_lib` (MTA directory path)

DELETED.

The directory containing MTA run-time libraries is no longer configurable and instead is now determined via the `SERVERROOT` environment variable, namely `SERVERROOT/lib/`.

40.1.3 MTA Tailor options: `imta_bin` (MTA directory path)

DELETED.

The directory containing executable binaries is no longer configurable, instead now being determined via the `SERVERROOT` environment variable, namely `SERVERROOT/lib/`.

40.1.4 MTA Tailor options: `imta_table` (MTA directory path)

DELETED.

The location of the MTA configuration directory is now determined using the `SERVERROOT` environment variable, namely `SERVERROOT/config/`. Note that with Unified Configuration, many fewer files comprise the MTA configuration than formerly, with legacy configuration, so this `SERVERROOT/config/` directory contains fewer files.

40.1.5 MTA Tailor options: `imta_d1` (MTA directory path)

DELETED.

Obsolete directory used by iMS 5.x for directory synchronization. No longer used.

40.1.6 MTA Tailor options: `imta_log` (MTA directory path)

DELETED.

Formerly, the `imta_log` [MTA Tailor option](#) specified the directory where the MTA log files are stored. This location is now determined from the `SERVERROOT` environment variable, as `SERVERROOT/log/`. Sites may change it through the use of a symbolic link.

40.1.7 `imta_tmp` Option

The legacy configuration `imta_tmp` [MTA Tailor option](#) has been replaced in Unified Configuration by `tmpdir`.

40.1.8 `imta_db_tmp` Option

The former `imta_db_tmp` [MTA Tailor option](#) used to be used to specify the MTA database temporary file location. The subdirectory name stopped being configurable as of Messaging Server 7.0-0.04 when the former `IMTA_DB_TMP` MTA Tailor option (available in JES MS 6.x versions, and typically set during installation of such versions to `/tmp/.mtadb/`) was removed, and instead as of Messaging Server 7.0 the MTA began using a hard-coded subdirectory of `.imtadb` underneath the (still-configurable) `imta_tmp` location. In Unified Configuration, the legacy configuration MTA Tailor option `imta_tmp` has been replaced by the `tmpdir` MTA (or base) option.

In other words: In JES MS 6.x, the MTA database temp file directory was fully configurable via the former `IMTA_DB_TMP` MTA Tailor option, and was typically set via that option to `/tmp/.mtadb/`. As of Messaging Server 7.0, the MTA database temp file directory was hard-coded as the `.imtadb` subdirectory underneath `imta_tmp` -- so typically `/tmp/.imtadb/`. As of Unified Configuration, `imta_tmp` has been replaced by the `tmpdir` MTA (or base) option, so the MTA database temp file directory location is `tmpdir-value/.imtadb/`.

40.1.9 imta_queue Option

The obsolete `imta_queue` [MTA Tailor option](#) located the directory for the MTA's store-and-forward message queues. This directory is now determined from the `SERVERROOT` environment variable. Sites may change it using either a symbolic link or using it as a file system mount point.

40.1.10 imta_help Option

Formerly, the location of the MTA help files was controllable by an [MTA Tailor option](#), `imta_help`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete; in particular, the location of MTA help files is no longer configurable.

40.1.11 imta_lang Option

The legacy configuration `imta_lang` [MTA Tailor option](#) has been replaced in Unified Configuration by the `langdir` MTA option, `mta.langdir`.

40.1.12 MTA Tailor options: imta_program (MTA directory path)

DELETED.

Formerly, the `imta_program` MTA Tailor option specified the location of the directory containing site-supplied executables for the pipe channel. This directory location is no longer configurable, and is now always located as `SERVERROOT/data/site-programs/`. If location customization is desired, use symbolic links.

40.2 File name MTA Tailor options

A number of [MTA Tailor file options](#) existed in legacy configuration to specify the name (and location) of various MTA configuration files. Such Tailor options are, by and large, obsolete as of Unified Configuration, when "rationalized", consistent names and locations are used unconditionally.

40.2.1 MTA Tailor options: `imta_option_file` (MTA file path)

DELETED.

Location of the (legacy configuration mode) `option.dat` file. No longer configurable; when legacy configuration is used, the `option.dat` file is located as `SERVERROOT/config/option.dat`.

With a Unified Configuration, the `option.dat` file is obsolete and no longer used; instead MTA options are stored in the Unified Configuration file, `config.xml`, and inspected or modified using the `msconfig` utility. Generally, the options formerly stored in `option.dat` correspond to [mta.option-name options](#) in Unified Configuration -- though with occasional exceptions for options instead at base level or which have been consolidated with other, general, Messaging Server options.

40.2.2 MTA Tailor options: `imta_system_filter_file` (MTA file path)

DELETED.

Location of the MTA-wide Sieve filter file. No longer configurable, it is now located as `SERVERROOT/config/imta.filter`.

40.2.3 MTA Tailor options: `imta_config_file` (MTA file path)

DELETED.

Name and location of the primary MTA configuration file, `imta.cnf`. The name and location of this file are no longer configurable.

In legacy configuration mode, it is `SERVERROOT/config/imta.cnf`. But in Unified Configuration, the contents of the legacy `imta.cnf` file have been incorporated into the Unified Configuration file `config.xml` file, located at `SERVERROOT/config/config.xml`.

40.2.4 MTA Tailor options: `imta_xml_config_file` (MTA file path)

DELETED.

The location of the Unified Configuration file, `config.xml`, is automatically determined using the `SERVERROOT` environment variable, namely `SERVERROOT/config/config.xml`.

40.2.5 MTA Tailor options: `imta_command_data` (MTA file path)

DELETED.

The location and name of compiled command line data built by the `clbuild` from the `pmdf.cld` file. This location and name are no longer configurable, instead being located via the `SERVERROOT` environment variable, namely `SERVERROOT/config/advanced/command_data`.

40.2.6 MTA Tailor options: `imta_config_data` (MTA file path)

DELETED.

The location and name of the compiled configuration built by the `cnbuild` utility. The location and name are no longer configurable, instead being located via the `SERVERROOT` environment variable, namely `SERVERROOT/config/advanced/config_data`.

40.2.7 MTA Tailor options: `imta_charset_data` (MTA file path)

DELETED.

The location and name of compiled character set data built by the `chbuild` utility. This information is no longer configurable, instead being located via the `SERVERROOT` environment variable, namely `SERVERROOT/config/advanced/charset_data`.

40.2.8 MTA Tailor options: `imta_alias_file` (MTA file path)

DELETED.

Name and location of the MTA alias file. In legacy configuration, the name and location became no longer configurable as of Messaging Server 7.0, being instead located via the `SERVERROOT` environment variable, namely `SERVERROOT/config/aliases`

In Unified Configuration, the MTA alias file is obsolete and instead the contents of the MTA alias file have been incorporated into the Unified Configuration file, `config.xml`, as [alias](#) settings.

40.2.9 MTA Tailor options: `imta_primary_log` (MTA file path)

DELETED.

In legacy configuration, the MTA Tailor option `imta_primary_log` specified the name and location of today's MTA message transaction log file. This is no longer configurable, and instead located via the `SERVERROOT` environment variable as `SERVERROOT/data/log/mail.log_current`.

40.2.10 MTA Tailor options: `imta_secondary_log` (MTA file path)

DELETED.

Name and location of yesterday's [MTA message transaction log file](#). No longer configurable, and instead located via the SERVERROOT environment variable as SERVERROOT/data/log/mail.log_yesterday.

40.2.11 MTA Tailor options: imta_tertiary_log (MTA file path)

DELETED.

Name and location of the [MTA's cumulative message transaction log file](#). No longer configurable, and instead located via the SERVERROOT environment variable as SERVERROOT/data/log/mail.log.

40.2.12 imta_primary_connection_log Option

Name and location of today's [MTA connection transaction log file](#). No longer configurable.

40.2.13 imta_secondary_connection_log Option

Name and location of yesterday's [MTA connection transaction log file](#). No longer configurable.

40.2.14 imta_tertiary_connection_log Option

Name and location of the [MTA's cumulative connection transaction log file](#). No longer configurable.

40.2.15 imta_name_content_file Option

Table of conversions by media type and file extensions. The name and location of this file is no longer configurable.

40.2.16 imta_forward_data Option

Forward address translation [text database file](#) which replaces the deprecated crdb form of forward database. The name and location of this file is no longer configurable.

40.2.17 imta_reverse_data Option

The DELETED `imta_reverse_database` [MTA Tailor option](#) had been used to specify the name and location of the reverse address translation [text database file](#) which replaces the deprecated crdb form of the reverse database. The name and location of this file are no longer configurable.

40.2.18 imta_general_data Option

[General text database file](#) which replaces the deprecated crdb form of the [general database](#). The name and location of this file is no longer configurable.

40.2.19 imta_alias_database Option

DELETED: Location no longer configurable. (The `imta_alias_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated `crdb` form of [MTA alias database](#).)

Various ways of storing [aliases](#) are supported, not necessarily in a "database".

40.2.20 `imta_reverse_database` Option

DELETED: Location no longer configurable. (The `imta_reverse_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated `crdb` form of [MTA reverse database](#).)

For continued use of a so-called "reverse database" in modern configurations, see the [use_text_databases](#) [MTA option](#).

40.2.21 `imta_forward_database` Option

DELETED: Location no longer configurable. (The `imta_forward_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated `crdb` form of [MTA forward database](#).)

For continued use of a so-called "forward database" in modern configurations, see the [use_text_databases](#) [MTA option](#).

40.2.22 `imta_general_database` Option

DELETED: Location no longer configurable. (The `imta_general_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated `crdb` form of [MTA general database](#).)

For continued use of a so-called "general database" in modern configurations, see the [use_text_databases](#) [MTA option](#).

40.2.23 `imta_domain_database` Option

DELETED: Location no longer configurable. (The `imta_domain_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated [domain database](#).)

40.2.24 `imta_ssr_database` Option

DELETED: Location no longer configurable. (The `imta_ssr_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated [Server Side Rules database](#) -- the database accessed when evaluating `ssr: URLs`.)

40.2.25 `imta_dn_to_id_database` Option

Legacy `dirsync` DN to ID database.

40.2.26 `imta_user_profile_database` Option

DELETED: Location no longer configurable. (The `imta_user_profile_database` [MTA Tailor option](#) had been used to specify the location and name of the deprecated [profile database](#) -- a database optionally used by [pipe channels](#).)

40.2.27 imta_charset_option_file Option

DELETED: Location no longer configurable. (The `imta_charset_option_file` [MTA Tailor option](#) had been used to specify the name and location of `chbuild`'s charset options file. This name and location are no longer configurable.)

40.2.28 imta_mapping_file Option

DELETED: This location is not longer configurable.

In legacy configuration, the mappings file is now always `SERVERROOT/config.mappings`. In Unified Configuration, the content of the mappings file has been incorporated into the XML configuration file, `config.xml`, as named [mapping groups](#).

40.2.29 imta_conversion_file Option

DELETED: This location is no longer configurable. (The `imta_conversion_file` [MTA Tailor option](#) had been used to specify the name and location of the conversions file storing message conversion commands. This name and location are no longer configurable.)

40.2.30 imta_hold Option

Formerly, the name and location of the file storing the list of stopped MTA [channels](#) (see the [qm utility's stop command](#)) was controllable by an [MTA Tailor option](#), `imta_hold`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete; in particular, the name and location of the file listing stopped MTA channels is no longer configurable.

40.2.31 imta_jbc_config_file Option

Formerly, the location of the Job Controller's configuration file was controlled by an [MTA Tailor option](#), `imta_jbc_config_file`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete, as the contents of the Job Controller configuration file have been incorporated into the XML configuration file; see instead [Job Controller options](#).

40.2.32 imta_dispatcher_config Option

Formerly, the location of the Dispatcher's configuration file was controlled by an [MTA Tailor option](#), `imta_dispatcher_config`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete, as the contents of the Dispatcher configuration file have been incorporated into the XML configuration file; see instead [Dispatcher options](#).

40.2.33 imta_libutil Option

Formerly, the name and location of the `imtautil` shared library was controllable by an [MTA Tailor option](#), `imta_libutil`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete; in particular, the name and location of the `imtautil` shared library is no longer configurable.

40.2.34 imta_libmap Option

Formerly, the name and location of the imtmap shared library was controllable by an [MTA Tailor option](#), `imta_libmap`. That Tailor option (indeed, the Tailor file as a whole) is now obsolete; in particular, the name and location of the imtmap shared library is no longer configurable.

40.2.35 imta_security_config_file Option

Formerly, the name and location of the MTA security configuration file was controllable by an [MTA Tailor option](#), `imta_security_config_file`. That MTA Tailor option has been deleted; the contents of the MTA security configuration file have been incorporated into the XML configuration file instead.

40.3 User MTA Tailor options

As of Messaging Server 7.0.5, the former [imta_user](#), [imta_user_username](#), and [imta_world_group](#) MTA Tailor options have been moved to the `restricted.cnf` file under the names `user`, `noprivuser`, and `group`, respectively.

40.3.1 Tailor file options: imta_user (string)

DELETED as of MS 7.0.5.

In earlier versions, the (now deleted) `imta_user` MTA Tailor option specified the user name for MTA operations. As of MS 7.0.5, it was moved to the `restricted.cnf` file as the "user" option (which also subsumes the former Message Store [serveruid](#) option).

40.3.2 Tailor file options: imta_user_username (string)

DELETED.

User name for untrusted pipe channel operations. Moved to the `restricted.cnf` file as the "noprivuser" option.

40.3.3 Tailor file options: imta_world_group (string)

DELETED.

Prior to MS 7.0.5, the `imta_world_group` MTA Tailor file option specified the group name for messaging server operations. As of MS 7.0.5, it has instead been moved to the `restricted.cnf` file as the "group" option.

40.4 Scheduling MTA Tailor options

Various former MTA Tailor options relating to scheduling of MTA operations have been moved (sometimes renamed) or obsoleted as of Messaging Server 7.0.5.

The former [imta_version_limit](#), [imta_version_limit_period](#), [imta_return_period](#), [imta_return_synch_period](#), and [imta_synch_cache_period](#) MTA Tailor options are obsolete. The effects they were used to achieve should instead nowadays be configured via the appropriate settings on relevant [Scheduler tasks](#), in particular, settings on the `purge` and `return_job` tasks, and by using the [synch_time Job Controller option](#).

The former `imta_return_split_period` has become `mta.return_split_period`; `imta_verify_return` has become (note the change in name!) `mta.return_verify`; `imta_return_split_period` has become `mta.return_split_period`; `imta_return_cleanup_period` has become `mta.return_cleanup_period`.

40.4.1 imta_version_limit Option

The `imta_version_limit` option is obsolete as of Messaging Server 7.0.5. (Formerly controlled the number of MTA log file versions retained when `imsimta purge` was run periodically.)

40.4.2 imta_version_limit_period Option

Obsolete option.

40.4.3 imta_return_synch_period Option

Obsolete option.

40.4.4 imta_return_split_period Option

The legacy configuration `imta_return_split_period` [MTA Tailor option](#) has been replaced in Unified Configuration by the `return_split_period` MTA option, `mta.return_split_period`.

40.4.5 imta_return_cleanup_period Option

The legacy configuration `imta_return_cleanup_period` [MTA Tailor option](#) has been replaced in Unified Configuration by the `return_cleanup_period` MTA option, `mta.return_cleanup_period`.

40.4.6 imta_verify_return Option

The legacy configuration `imta_verify_return` [MTA Tailor option](#) has been replaced in Unified Configuration by the `return_verify` MTA option, `mta.return_verify`.

40.4.7 imta_synch_cache_period Option

Obsolete option.



Chapter 41 Dispatcher

41.1 Dispatcher operation	41-1
41.1.1 Creation and expiration of Dispatcher Worker Processes	41-2
41.2 Dispatcher options	41-2
41.2.1 enable Option Under dispatcher	41-3
41.2.2 backlog Option Under service	41-3
41.2.3 debug Option Under dispatcher	41-3
41.2.4 dns_verify_domain Dispatcher option	41-3
41.2.5 historical_time Dispatcher option	41-4
41.2.6 image Dispatcher option	41-4
41.2.7 interface_address Dispatcher or Job Controller option	41-4
41.2.8 listenaddr Dispatcher (global) option	41-4
41.2.9 listenaddr Dispatcher (service) option	41-5
41.2.10 logfile_name Option	41-5
41.2.11 max_conns Dispatcher option (non-negative integer)	41-5
41.2.12 Dispatcher service options: max_conns (non-negative integer)	41-6
41.2.13 min_conns Dispatcher option	41-6
41.2.14 max_handoffs Dispatcher option	41-7
41.2.15 max_idle_time Option	41-7
41.2.16 max_life_conns Option	41-7
41.2.17 max_life_time Option Under dispatcher	41-7
41.2.18 max_procs Dispatcher option	41-7
41.2.19 max_shutdown Option	41-8
41.2.20 min_procs Dispatcher option	41-8
41.2.21 parameter Dispatcher option	41-8
41.2.22 port Option Under dispatcher	41-8
41.2.23 stacksize Option	41-8
41.2.24 ssl_ports Dispatcher option	41-9
41.2.25 tcp_ports Option	41-9
41.2.26 user Option Under dispatcher	41-10
41.2.27 user Option Under service	41-10
41.2.28 tls_min_bits Dispatcher service option	41-10
41.2.29 tls_bits_reject_msg Dispatcher service option	41-10
41.2.30 use_nslog Option Under dispatcher	41-10
41.2.31 Old Dispatcher options	41-11

The Dispatcher is one of the two major, "control" processes of the MTA (the other major such process being the [Job Controller](#)).

The MTA's Dispatcher is a multithreaded connection dispatching agent that permits multiple multithreaded server processes to share responsibility for a given service. When using the MTA's service Dispatcher, it is possible in particular to have several multithreaded SMTP server processes and several SMTP SUBMIT server processes running concurrently. In addition to having multiple server processes for a single service, each server process may have one or more active connections.

41.1 Dispatcher operation

The Dispatcher works by acting as a central receiver for the [TCP ports](#) listed in its configuration. For each defined service, the Dispatcher may create one or more Worker Processes that will actually handle the connections after they've been established.

The Dispatcher can selectively accept or reject incoming connections to the services it manages. See the [PORT_ACCESS](#) mapping table.

In general, when the Dispatcher receives a connection for a defined TCP port, it checks its pool of available Worker Processes 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 proactively 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 [Creation and expiration of Dispatcher Worker Processes](#) and [Dispatcher options](#).

41.1.1 Creation and expiration of Dispatcher Worker Processes

There are automatic housekeeping facilities within the Dispatcher to control the creation of new and expiration of old or idle Worker Processes. The basic options that control the Dispatcher's behavior in this respect are [min_procs](#) and [max_procs](#). [min_procs](#) provides a guaranteed level of service by having a number of Worker Processes ready and waiting for incoming connections. [max_procs](#), on the other hand, sets an upper limit on how many Worker Processes may be concurrently active for the given service.

Since it is possible that a currently running Worker Process might not be able to accept any connections either 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 Worker Processes. [min_conns](#) specifies the number of connections that flags a Worker Process as "busy enough" while [max_conns](#) specifies the "busiest" that a Worker Process can be.

In general, the Dispatcher will create a new Worker Process when the current number of Worker Processes is less than [min_procs](#) or when all existing Worker Processes are "busy enough" (the number of currently active connections each has is at least [min_conns](#)).

The [max_life_conns](#), [max_life_time](#), and [max_idle_time](#) Dispatcher options all affect when the Dispatcher will expire "old" or idle Worker Processes.

Note that if a Worker Process is killed unexpectedly, *e.g.*, by the UNIX `kill` command, the Dispatcher will still create new Worker Processes as new connections come in. Only shutting down the specific Dispatcher service, disabling the specific Dispatcher service followed by restarting the Dispatcher, or shutting down the Dispatcher itself, will stop the Dispatcher's creation of new, as-needed, Worker Processes, (up to the Dispatcher's configured maximum number of such processes). (See the [shutdown](#) utility, or after disabling a service the [restart](#) utility, for how to shut down a service entirely.)

41.2 Dispatcher options

The Dispatcher has a number of options, settable either at the Dispatcher top level (hence affecting overall Dispatcher operation or becoming a default for Dispatcher services), or settable under specific Dispatcher services (hence applying only to that service). *E.g.*,

```
msconfig> set dispatcher.debug 7
msconfig# set dispatcher.min_procs 2
msconfig# set dispatcher.service:SMTP.min_procs 3
```

See also the [logfile options](#) set as `dispatcher.logfile.*`.

In addition to the substantial number of Dispatcher options relevant to modern configurations, the Dispatcher also has a very large number of older options of only historical interest nowadays (such as those for setting various OpenVMS process quotas), listed under [Old Dispatcher options](#).

41.2.1 enable Option Under dispatcher

The `enable Dispatcher option` is used to enable the [dispatcher daemon](#). This defaults to the value of the `mta.enable` (Unified Configuration) or `local.imta.enable` (legacy configuration) option. When the Dispatcher is enabled, the default for `schedule.task:purge.enable` (Unified Configuration) or `local.schedule.purge.enable` (legacy configuration) is 1.

41.2.2 backlog Option Under service

The `backlog Dispatcher service option` controls the depth of the TCP backlog queue for the socket. The default value for each Dispatcher service is `max_conns*max_procs` for that service (with a minimum value of 128). Prior to the 7.0.5 release, the minimum value was 5.

41.2.3 debug Option Under dispatcher

The `debug Dispatcher option` enables debug output from the Dispatcher.

41.2.4 dns_verify_domain Option

Various groups maintain information about spam sources or open relay sites and some sites like to check incoming IP connections against the lists maintained by such groups. The `dns_verify_domain Dispatcher option` specifies the host name or IP address of source against which to check incoming connections.

Note that [PORT_ACCESS mapping table](#) probes are made before any `dns_verify_domain` lookups are consulted. If a `PORT_ACCESS` probe rejects a connection, then the `dns_verify_domain` lookup does not need to be (and is not) performed. And as of JES MS 6.0, an explicit match in the `PORT_ACCESS` mapping table that accepts a connection will cause any `dns_verify_domain` lookups to be omitted for that connection; thus the `PORT_ACCESS` mapping table can be used to "white list" source IP addresses (such as internal IP addresses) which should not receive the DNS verification lookup.

In legacy configuration, up to five `dns_verify_domain` options are permitted for each service. In Unified Configuration, the `dns_verify_domain Dispatcher service option` takes a host-list of up to five hosts. (Note that SMTP is typically the only service for which such checks make sense.) For example, in Unified Configuration:

```
msconfig> set dispatcher.service:SMTP.dns_verify_domain "rbl.maps.vis.com dul.maps.vis.com"
```

Or analogously in legacy configuration:

```
[SERVICE=SMTP]
PORT=25
DNS_VERIFY_DOMAIN=rbl.maps.vix.com
DNS_VERIFY_DOMAIN=dul.maps.vix.com
```

If this option is enabled on a well-known port (25, 110, or 143), then a standard message such as the one below will be sent before the connection is closed:

```
500 5.7.1 access_control: host 192.168.51.32 found on DNS list and rejected
```

If you wish the MTA to log such rejections, you may set the 24th bit (starting at bit 0) of the Dispatcher debugging [debug](#) option, in Unified Configuration:

```
msconfig> set dispatcher.debug 16777216
```

or in legacy configuration `debug=16%1000000`, to cause logging of the rejections to the `dispatcher.log` file; see Section 10.6. Such `dispatcher.log` entries will take the form:

```
access_control: host a.b.c.d found on DNS list and rejected
```

41.2.5 historical_time Option

The `historical_time` Dispatcher option controls how long (in seconds) expired connections (those that have been closed) and processes (those that have exited) remain listed for statistical purposes. Note that the setting of this option affects the amount of virtual memory that the Dispatcher requires.

The default has been 120 (2 minutes), but initial configuration set this value to 0 (no history retained). Prior to 7.0.5, the default value was 120 (2 minutes) but initial configuration set this to 0 (no history). As of 7.0.5, the default value is 0.

41.2.6 image Option

The `image` Dispatcher option specifies the binary executable file that will be run by Worker Processes when such processes are created by the Dispatcher. Note that the specified executable file should be one designed to be controlled by the Dispatcher. Service names beginning with "SMTP", "LMTP" or "MSADMIN" have default paths built-in as of the 7.0.5 release.

41.2.7 interface_address Option

The legacy option `interface_address` (Job Controller option and Dispatcher option) is an alias for the [listenaddr](#) option in Unified Configuration.

41.2.8 listenaddr Option Under dispatcher

The `listenaddr` Dispatcher option (formerly `INTERFACE_ADDRESS` in legacy configuration) can be used to specify the IPv4 address interface to which the Dispatcher service should bind. By default, the Dispatcher binds to all IP addresses. But for systems having multiple network interfaces each with its own IP address, it may be useful to bind

different services to the different interfaces. `listenaddr` may be set either directly under `dispatcher` as `dispatcher.listenaddr`, in which case it sets the global default for all services, or may be set under a specific service, `dispatcher.service:service-name.listenaddr`, in which case it is setting the interface address to which that particular service should bind.

Note that if `listenaddr` is specified for a service, then that is the only interface IP address to which that Dispatcher service will bind. Only one such explicit interface IP address may be specified for a particular service (though other similar Dispatcher services may be defined for other interface IP addresses). Note that the [interfaceaddress](#) channel option provides the complementary capability for specifying which interface address a TCP/IP channel uses for outgoing connections and messages.

The allowed values for this option include an IPv4 address in dotted decimal form (e.g., 127.0.0.1), or a short form or fully-qualified DNS host name which will be resolved to an IPv4 address by obtaining the DNS A record for the name. To explicitly specify the default value of binding to all available interfaces, the string "INADDR_ANY" may be used. To bind to the loopback device, 127.0.0.1, the string "localhost" may be specified.

41.2.9 listenaddr Option Under service

The `listenaddr` Dispatcher option (formerly `INTERFACE_ADDRESS` in legacy configuration) can be used to specify the IPv4 address interface to which the Dispatcher service should bind. By default, the Dispatcher binds to all IP addresses. But for systems having multiple network interfaces each with its own IP address, it may be useful to bind different services to the different interfaces. `listenaddr` may be set either directly under `dispatcher` as `dispatcher.listenaddr`, in which case it sets the global default for all services, or may be set under a specific service, `dispatcher.service:service-name.listenaddr`, in which case it is setting the interface address to which that particular service should bind.

Note that if `listenaddr` is specified for a service, then that is the only interface IP address to which that Dispatcher service will bind. Only one such explicit interface IP address may be specified for a particular service (though other similar Dispatcher services may be defined for other interface IP addresses). Note that the [interfaceaddress](#) channel option provides the complementary capability for specifying which interface address a TCP/IP channel uses for outgoing connections and messages.

The allowed values for this option include an IPv4 address in dotted decimal form (e.g., 127.0.0.1), or a short form or fully-qualified DNS host name which will be resolved to an IPv4 address by obtaining the DNS A record for the name. To explicitly specify the default value of binding to all available interfaces, the string "INADDR_ANY" may be used. To bind to the loopback device, 127.0.0.1, the string "localhost" may be specified.

41.2.10 logfilename Option

Specifying the `logfilename` option for a service causes the Dispatcher to direct output for corresponding Worker Processes to the specified file. The log file may include the system's name by including the `%s` token. The default log file name is the service name with a ".log" suffix, except for "SMTP_SUBMIT" which defaults to "IMTA_LOG:tcp_submit_server.log" and "SMTP" which defaults to "IMTA_LOG:tcp_smtp_server.log".

41.2.11 max_conns Option Under dispatcher

Dispatcher service options:
`max_conns` (non-negative integer)

The `max_conns` Dispatcher option specifies the maximum number of concurrent connections handled by a single server process (Worker Process). When the maximum number of concurrent sessions is reached, the server process stops listening for new connections. When all currently open connections are closed, the process will exit. Setting `dispatcher.max_conns` sets a global default, which may be overridden for specific services by setting `max_conns` under that service, e.g., `dispatcher.service:service-name.max_conns`.

Prior to Messaging Server 7.0.5, the default value was 10, but initial configuration set the option explicitly to 50. As of 7.0.5, the default value is now 50. In Messaging Server 7.2 and earlier, the maximum possible value for this option was unconditionally 50, with attempts to set a value higher than 50 resulting in the value of 50 being used. As of Messaging Server 7.3, the maximum of 50 is only enforced on 32 bit platforms; 64 bit platforms allow any value.

Setting `max_conns` to higher values allows more connections, but at the potential cost of decreased performance for each individual connection. If it is set to 1, then for every incoming client connection, only one server process will be used. When the client shuts down, the server process will also exit. Note that the `max_conns` value times the `max_procs` value controls the maximum number of simultaneous connections that can be accepted.

41.2.12 `max_conns` Option Under service

The `max_conns` Dispatcher service option, `dispatcher.service:service-name.max_conns`, specifies the maximum number of concurrent connections handled by a single server process (Worker Process) for this service. When the maximum number of concurrent sessions is reached, the server process stops listening for new connections. When all currently open connections are closed, the process will exit.

Such a per-service setting overrides any global Dispatcher setting (`dispatcher.max_conns`).

The default value for this option has been 10. As of 7.0.5, the new default is 50. In Messaging Server 7.2 and earlier, the maximum possible value for this option was unconditionally 50, with attempts to set a value higher than 50 resulting in the value of 50 being used. As of Messaging Server 7.3, the maximum of 50 is only enforced on 32 bit platforms; 64 bit platforms allow any value.

In principle, for services running under the Dispatcher where the server image is not multithreaded, this option must be set to 1. However, the main Dispatcher services of interest, including the MTA's [SMTP server](#) and [LMTP server](#), are multi-threaded and therefore capable of handling multiple clients. For such multithreaded servers, the choice of setting for this option is mainly a performance issue relating to the number of processes and the size of the process virtual address space. Setting `max_conns` to higher values allows more connections, but at the potential cost of decreased performance for each individual connection. If it is set to 1, then for every incoming client connection, only one server process will be used. When the client shuts down, the server process will also exit. Note that the `max_conns` value times the `max_procs` value controls the maximum number of simultaneous connections that can be accepted.

41.2.13 `min_conns` Option

The Dispatcher attempts to distribute connections evenly across its pool of currently available Worker Processes. The Dispatcher uses the `min_conns` value to determine the

minimum number of connections that each Worker Process must have before there will be any consideration of adding new Worker Processes to the pool. Prior to 7.0.5, the default value was 3 while initial configuration set this to 30. As of 7.0.5, the default value is 30.

min_conns may be set either under `dispatcher` (to establish a general default for Dispatcher services), or under a specific named `service` (to apply only to that named service).

41.2.14 max_handoffs Option

The `max_handoffs` Dispatcher option specifies the maximum number of concurrent asynchronous handoffs in progress that the Dispatcher will allow for newly established TCP/IP connections to a service port. The default value is 5.

41.2.15 max_idle_time Option

When a Worker Process has had no active connections for the period of the `max_idle_time` Dispatcher option, the process will be eligible for being shut down. `max_idle_time` may be set either directly under `dispatcher`, to set a default for all Dispatcher services, or may be set for a particular service under the Dispatcher, *e.g.*, `dispatcher.service:SMTP.max_idle_time`. Note that the `max_idle_time` option is only effective if there are more than the value of `min_procs` Worker Processes currently in the Dispatcher's pool for this service.

The default value has been 0, while initial configuration set this to 600. Prior to 7.0.5, the default value was 0 while initial configuration set this to 600. As of 7.0.5, the default value is 600 seconds (10 minutes).

41.2.16 max_life_conns Option

As part of the Service Dispatcher's ability to perform Worker Process housekeeping, the `max_life_conns` Dispatcher option requests that Worker Processes only be kept around for the specified number of connections. After a Worker Process has handled the specified number of connections, it is subject to being shut down. For instance, when specified in an SMTP service section, this is the number of total connections the SMTP server is able to accept before being retired so a new SMTP server process can take its place. This is different from the `max_conns` Dispatcher option, which limits the number of concurrent connections.

The default value has been 300, while initial configuration set the option value to 10000. As of 7.0.5, the default value is 10000.

For the Job Controller, the legacy configuration `max_life_conns` option has been renamed to `max_life_askwork`.

41.2.17 max_life_time Option Under dispatcher

When the Dispatcher creates a server Worker Process, a countdown timer is set to the number of seconds specified by the `max_life_time` Dispatcher option. When this countdown time has expired, the Worker Process is subject to being shut down. The Dispatcher's default is 1 day (86400).

41.2.18 max_procs Option

The `max_procs` Dispatcher option controls the maximum number of Worker Processes that will be created for this service. Thus this value times `max_conns` specifies the maximum number of simultaneous connections that can be handled. Prior to 7.0.5, the default value was 5 while initial configuration set this to 10. As of 7.0.5, the default value is 10.

41.2.19 max_shutdown Option

The `max_shutdown` Dispatcher option specifies a maximum number of processes that may be shut down. In order to provide a minimum availability for the service, the Service Dispatcher will not shut down Worker Processes that might otherwise be eligible to be shut down if shutting them down would result in having more than `max_shutdown` Worker Processes for the service in the shutting down state. This means that processes that may be eligible to be shut down may continue running until a shutdown "slot" is available. Having this option set to about half of the value of the `max_procs` option is usually appropriate. Prior to 7.0.5, the default value was 2. As of 7.0.5, the default value is 5.

41.2.20 min_procs Option

The `min_procs` Dispatcher option determines the minimum number of Worker Processes that will be created by the Dispatcher for the current service. Upon startup, the Dispatcher will create this many detached processes to start its pool. When an old such process expires, the Dispatcher will ensure that there are at least this many available processes (by creating a new process, if necessary) in the pool for this service.

As of 7.0.5, the default is 1; in prior versions, the default had been 0.

41.2.21 parameter Option

The interpretation and allowed values for the `parameter` Dispatcher service option are service specific. In the case of an SMTP service, the `parameter` option may be set to `CHANNEL=channelname`, to associate a default TCP/IP channel with the port for that SMTP service. (Note that the presence of the equal sign within the value means that when using `msconfig`, the value will need to be quoted.) For example, the `SMTP_SUBMIT` service sets `parameter` to "`CHANNEL=tcp_submit`" so that incoming mail submissions are handled on that channel.

```
msconfig> show service:SMTP_SUBMIT.parameter
role.dispatcher.service:SMTP_SUBMIT.parameter = CHANNEL=tcp_submit
```

Note that this can be useful if you wish to run SMTP servers on multiple ports---perhaps because your internal POP and IMAP clients have been configured to use a port such as the submission port 587 rather than port 25, thus separating their message traffic from incoming SMTP messages from external SMTP hosts---and if you wish to associate different TCP/IP channels with the different port numbers.

41.2.22 port Option Under dispatcher

In Unified Configuration, the `tcp_ports` Dispatcher service option is the preferred name for the Dispatcher PORT option used in legacy configuration.

41.2.23 stacksize Option

The `stacksize` Dispatcher service option specifies a minimum per-thread stack size. Various components may have their own minimum values; the larger of an explicitly specified `stacksize` option value and the component's own internal minimum will be used. The default value is 2048000 as of the 7.0.5 release.

41.2.24 `ssl_ports` Option Under service

The `ssl_ports` Dispatcher option specifies the TCP port(s) on which the Dispatcher will listen for incoming TLS connections for the current service. Connections made to this port or these ports will automatically (without use of commands such as `STARTTLS` in the case of SMTP) negotiate TLS use and be transferred to one of the Worker Processes created for this service. The SMTP client analogue of this option is the (new in 7.0.5) `SSL_CLIENT` TCP/IP-channel-specific option.

See also the `tcp_ports` Dispatcher service option.

41.2.25 `tcp_ports` Option

The `tcp_ports` option specifies a list of regular (non-SSL) ports on which this service listens for connections. For services which instead only support listening on a single (non-SSL) port, see instead the `port` option.

Some services also support listening on (dedicated) SSL ports; for those services, see also the `ssl_ports` option.

41.2.25.1 Use with service

The `tcp_ports` Dispatcher service option (which is the new, Unified Configuration name for the legacy configuration `PORT` Dispatcher option) specifies the TCP port(s) on which the Dispatcher will listen for incoming connections for the current service. Connections made to this port or these ports will be transferred to one of the Worker Processes created for this service.

For instance, in a typical configuration:

```
msconfig> show dispatcher.service:SMTP.tcp_ports
role.dispatcher.service:SMTP.tcp_ports = 25
msconfig> show dispatcher.service:SMTP_SUBMIT.tcp_ports
role.dispatcher.service:SMTP_SUBMIT.tcp_ports = 587
```

meaning that the Dispatcher runs the SMTP service on port 25, and the SMTP SUBMIT service on port 587.

See also the `ssl_ports` Dispatcher service option.

41.2.25.2 Use with `smpp_relay`

The `tcp_ports` SMS `smpp_relay` option specifies the TCP port for inbound SMPP client connections. Specification of this option is mandatory; there is no default value. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

41.2.25.3 Use with `smpp_server`

The `tcp_ports` `SMS` `smpp_server` option specifies the TCP port for inbound SMPP server connections. Specification of this option is mandatory; there is no default value. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

41.2.25.4 Use with job controller

The `tcp_ports` Job Controller option, `job_controller.tcp_ports`, specifies the TCP port on which the Job Controller should listen for request packets; that is, it is the port on which the Job Controller listens for its internal protocol communications. You generally do not want to change this option unless the default conflicts with another TCP application on your system. This is a global Job Controller option, set directly under `job_controller`; it is not available under channel or pool groups. The default is 27442.

41.2.25.5 Use with tcp_listen

Under a top level `imapproxy`, `popproxy`, or `submitproxy` component, inside a named `tcp_listen` group, the `tcp_ports` option specifies the TCP port(s) on which that proxy server listens; *e.g.*:

```
msconfig> set imapproxy.tcp_listen:imap-proxy-1.tcp_ports 143
```

41.2.26 user Option Under dispatcher

The `user` Dispatcher option used to specify the Unix user id to run the Dispatcher worker processes as. This option is now ignored.

41.2.27 user Option Under service

The `user` Dispatcher service option used to specify the Unix user id to run the Dispatcher worker processes as. This option is now ignored.

41.2.28 tls_min_bits Option

The `tls_min_bits` Dispatcher service option specifies the minimum number of bits of encryption strength that must be in use in order for the connection to be permitted. Connections that are established with fewer bits of encryption, including no encryption, will be sent the error text defined by `tls_bits_reject_msg`, and then closed.

41.2.29 tls_bits_reject_msg Option

The `tls_bits_reject_msg` Dispatcher service option specifies some optional error text to send before the connection is closed when a connection fails to meet the minimum encryption strength required by `tls_min_bits`.

41.2.30 use_nslog Option Under dispatcher

The `use_nslog` Dispatcher option may be set to 1 to enable use of `nslog()` for Dispatcher debug log files. The default is 0. When `use_nslog` has been enabled, see also the [logfile options](#) that may be set as `dispatcher.logfile.*`. Note that the `loglevel` option is

not supported for the [Dispatcher](#) as its debug level is controlled instead by use of the [debug](#) Dispatcher option, `dispatcher.debug`.

41.2.31 Old Dispatcher options

Many Dispatcher options are no longer relevant for the modern MTA. See older printed documentation for details on these Dispatcher options.

Such historical Dispatcher options include:

- `astlm`
- `biolm`
- `bytlm`
- `cpulm`
- `diolm`
- `enqlm`
- `fillm`
- `group`
- `jtquota`
- `pgflquota`
- `prcmlm`
- `tqelm`
- `wsdefault`
- `wsextent`
- `wsquota`
- `trace_total_buffer_size`
- `trace_per_conn_buffer_size`
- `enable_rbl`
- `ident`
- `vms_group`
- `ucx_hold`
- `process_priority`
- `needs_dcl`
- `set_network`
- `new_features`
- `wp_timeout`
- `unix_domain`
- `unix_domain_dir`



Chapter 42 Job Controller

42.1 Job Controller operation	42-2
42.1.1 Job Controller operation under stress	42-3
42.1.2 Job Controller priority-based processing	42-4
42.2 Job Controller default configuration	42-6
42.3 Job Controller options	42-9
42.3.1 enable Option Under job controller	42-10
42.3.2 debug Option Under job controller	42-10
42.3.3 listenaddr Job Controller option	42-10
42.3.4 job_limit Option	42-11
42.3.5 master_command Option	42-11
42.3.6 max_cache_messages Option	42-12
42.3.7 max_life_askwork Option	42-13
42.3.8 max_life_time Option Under job controller	42-13
42.3.9 notice_time Option	42-13
42.3.10 port Option Under job controller	42-13
42.3.11 rebuild_parallel_channels Option	42-13
42.3.12 secret Option Under job controller	42-14
42.3.13 slave_command Option	42-14
42.3.14 stressblackout Option	42-14
42.3.15 stresstime Option	42-14
42.3.16 stressfactor Option	42-14
42.3.17 unstressfactor Option	42-15
42.3.18 stressjobs Option	42-15
42.3.19 unstressjobs Option	42-15
42.3.20 synch_time Option	42-15
42.3.21 tcp_ports Option Under job controller	42-15
42.3.22 nonurgent_delivery, normal_delivery, urgent_delivery Job Controller options	42-15
42.3.23 use_nslog Option Under job controller	42-16
42.3.24 job_pool	42-17
42.3.25 channel_class	42-17
42.4 Checking that the Job Controller is running	42-18

The Job Controller is one of the two major, "control" processes of the MTA (the other major such process being the [Dispatcher](#)).

The Job Controller has two main responsibilities: (1) maintaining an in-memory "queue cache database" containing information about what message files are on disk awaiting delivery; and (2) scheduling and executing [channel](#) jobs to perform message processing (attempt message delivery). (The Job Controller is capable also of running additional, non-channel, periodically scheduled jobs. But that capability is not normally used in modern versions of the MTA.)

In contrast to the [Dispatcher](#), which is responsible for accepting incoming TCP/IP connections and creating and managing server processes, the Job Controller is responsible for creating and managing MTA channel jobs (in particular SMTP client outbound processes and Message Store delivery channel processes) to attempt message delivery. Notable examples are that the Dispatcher oversees SMTP server processes for accepting messages incoming to the MTA, whereas the Job Controller oversees Message Store delivery channels and SMTP channel client processes for delivering outbound messages. (Though note that there can be exceptions to the overly simplistic "inbound equals Dispatcher, delivery equals Job Controller" separation, as in the case of channels that "pull" messages as well as deliver messages, or in the case of

delivery channels that also enqueue new messages such as [notification messages](#).) The topic [Job Controller operation](#) will go further into Job Controller operation.

As the Job Controller is so fundamental to MTA operation, in normal operation it should always be present; see the topic [Checking that the Job Controller is running](#). Indeed, normally the Watcher and msprobe are configured to monitor and perform automatic checks on the Job Controller, with the Watcher restarting the Job Controller if it appears to be absent or malfunctioning; see Section 29.4.2.

Furthermore, as operational advice, note that other than in cases of drastic configuration changes to the MTA or to site deployment, when a Job Controller [restart](#) often is necessary in order for changes to take effect, normally the Job Controller should be left running, without gratuitous restarting, as shutting down the delivery "half" of the MTA even briefly tends to be disruptive to optimal performance of outbound message delivery. (In particular, message delivery problems or delivery throughput concerns are not good reasons to restart the Job Controller! Message delivery problems should be attacked at the channel level where the delivery problem actually occurs, not at the Job Controller level; and overall delivery throughput generally suffers a temporary dip when the Job Controller must be restarted.) Certain operationally interesting Job Controller options can instead have values adjusted "on the fly" (without requiring a Job Controller restart) using the `imsimta cache -change` utility.

42.1 Job Controller operation

The Job Controller does not process or deliver messages itself, but rather keeps track of message files, and creates and manages channel jobs to process those messages.

Upon receipt of an incoming message from any source, the MTA [channel](#) that is handling the receipt of the message determines the destination, enqueues the message, and sends a request to the Job Controller to execute the next channel. The Job Controller will then initiate a channel job, if one is needed (that is, if there is not such a channel job already running, or if there are not "enough" jobs for that channel). Channel jobs, in turn, ask for and receive from the Job Controller the name of which message they should process next. When there are multiple messages to process, a channel process may end up running for "awhile", in a cycle of [asking the Job Controller for a message](#) and then processing (delivering) it, and then asking for another, *etc.* And if the number of messages eligible for immediate delivery attempts is sufficiently "high", the Job Controller will initiate more than one channel job to work in parallel, each delivering a subset of the messages.

Internally, the Job Controller maintains a data structure of a set of queues of messages awaiting delivery attempts. New messages are inserted into this data structure sorted onto queues according to their destination channel, in some cases also sorted by their destination domain name, and further sorted according to [message processing priority](#). Additional queues are maintained (one for each destination channel) of those messages that have already had at least one unsuccessful delivery attempt, and which are waiting for another delivery attempt.

The Job Controller configuration establishes processing [pools](#); each such pool has a limit (`job_limit`) on how many processes may execute in it simultaneously (and a pool may optionally be configured with restrictions on times of day or days of the week in which it may execute processes). Each channel is constrained (via the `pool` channel option) to run in one such pool, and may optionally be further constrained on how many processes it may run simultaneously within the pool (`max_jobs`). Multiple channels may be configured to run in the same pool, if it is desired for those channels to share (contend for) the same pool of process slots.

The Job Controller tracks how many processes each channel has running (and in the case of multithreaded channels specifically written to operate with the Job Controller by letting the Job Controller initiate delivery threads, the Job Controller tracks how many threads are running). When there are "enough" messages eligible for an immediate delivery attempt, the Job Controller will initiate a new delivery thread, or whole new delivery process, as needed (if the configured limits on such jobs have not yet been reached).

The Job Controller will also "cycle" channel jobs, aging out (expiring) sufficiently old channel jobs and then creating new channel jobs (as needed) to take their place. Thus channel jobs, even for busy channels, have a limited life-span. That this is a built-in aspect of the Job Controller both increases robustness in the face of unexpected problems, and ensures that updates to the MTA configuration, and changes to user and domain data in LDAP, will propagate through to affect channel jobs automatically, with bounded delay.

As part of the Job Controller's housekeeping and self-maintenance of its internal message queueing data structures, the Job Controller will periodically do a disk scan of the MTA queue area, to detect any message files omitted from its in-memory queues and reconcile its in-memory lists with what is physically present on disk.

So to summarize: the Job Controller's primary responsibilities are to maintain internal queues of which messages need delivery attempts and when, to initiate channel jobs to attempt those deliveries as needed, and to hand over to channel jobs the name of which message the job should attempt to process next.

42.1.1 Job Controller operation under stress

The Job Controller has several self-managing features that work together to aid in managing work load in general, and in particular to continue to operate successfully even under exceptionally heavy load. Beyond the Job Controller's general queueing strategy and creation (and expiration) of Worker Process threads discussed under [Job Controller operation](#), the Job Controller's `max_cache_messages` and `stress*` options particularly relate to operation under load.

Under typical circumstances, the Job Controller keeps track in memory of all the active messages (non-HELD message files) in the MTA queues. However, to limit its maximum memory requirement, the Job Controller has a configurable limit, `max_cache_messages`, on how many messages to track *in memory*. When the Job Controller's `max_cache_messages` capacity is exceeded, the Job Controller will not bother to retain *in memory* information about additional messages. But in such a case, the affected (excess) message files themselves had already been safely deposited in the MTA's store and forward message queues, where they will be detected later (at which time normal message processing will resume). At such times of "excess" messages, any enqueueing channel requests to the Job Controller for *immediate* insertion of messages into the Job Controller's "queue cache database" list of messages due for processing are ignored; but it is merely the *immediate* message processing that is suspended at such times for the affected messages. The Job Controller will detect any such "excess" message files later, during one of its housekeeping operations, and then begin delivery attempts for such messages. So `max_cache_messages` is a limit on how many messages will get normal/optimal processing (as in more-or-less immediate processing, in a "first in, first out" order); but messages that exceed `max_cache_messages` will be processed also, eventually.

New in Messaging Server 7.0 is a "stress" feature in the Job Controller, whereby the Job Controller can be informed that channels are "stressed" and then in response temporarily reduce delivery jobs on that channel to give the destination some respite. This is primarily relevant for Message Store delivery channels. Besides accepting new messages, the Message Store is also responsible for responding to end user e-mail client message access requests. So

when the Message Store is extraordinarily busy, temporarily reducing the rate of new message deliveries to the Message Store may allow the Message Store to instead focus its resources on maintaining responsiveness to end users; that is, slowing down insertion of new messages into the Message Store may free up the Message Store to respond more quickly to e-mail client access to existing mailboxes and messages.

Message Store delivery channels (`ims-ms` or `tcp_lmtpcs*` channels) will automatically inform the Job Controller of stress, when the Message Store detects that it is stressed. In the case of `ims-ms` channels, when an `ims-ms` channel job is about to shut down, it will query the Message Store as to whether the store is stressed, and if it is then that `ims-ms` channel job will inform the Job Controller. In the case of LMTP delivery, after each successful delivery into the Message Store the `LMTP server` will query as to whether the store is stressed and if so, the LMTP server will report that back to the `LMTP client` via a special

250 2.3.99 Delivery OK but store under stress

success status; the MTA's LMTP client recognizes that special status and will then inform the MTA Job Controller of the back end Message Store stress.

An MTA administrator may also, using the new `imsimta qm stress` command, manually direct the Job Controller to consider any arbitrary channel to be "stressed".

When the Job Controller is informed that a channel is under "stress", it checks to see if it has already been told this recently: the Job Controller ignores "stress" alerts that are received within `stressblackout` seconds of a previous stress alert for the same channel. But if the stress alert is "new" information, then the Job Controller will multiply the effective `threaddepth` parameter for the channel by `stressfactor`, and subtract `stressjobs` from the effective job limit for the channel. (In the absence of stress, the effective job limit would be simply the minimum of the channel's `maxjobs` and the `job_limit` for the pool in which that channel runs.¹) In addition, the Job Controller will ask all current master program processes for the channel to exit, and will, if the message queue for that channel is not empty, start an appropriate number of new processes: that is, any old processes are shut down and an appropriate, reduced number of new processes are started in their place.

But the Job Controller's cut back on jobs for "stressed" channels is intended to be temporary, on the presumption/hope that such a channel should/may "recover" after a time and be able to return to normal processing levels. The Job Controller attempts to gradually return to the originally configured settings, at step times and step sizes controlled by `stresstime` and `unstressfactor` and `unstressjobs`, as follows, (assuming that no further "stress" alerts or manual `imsimta qm stress` changes are received to further modify the settings and schedule). Automatically, `stresstime` seconds after the last stress change (or alternately upon receipt of an `imsimta qm unstress` command), the Job Controller divides the effective `threaddepth` by `unstressfactor` (never allowing the effective `threaddepth` to drop below the original configured `threaddepth`), and adds `unstressjobs` to the effective job limit (never allowing the effective job limit to rise above the original configured limit). A "stress change" is either an increase in stress or a decrease in stress.

Note¹ The effective `threaddepth` never goes over 134,217,727, and the effective job limit never goes below 1.

42.1.2 Job Controller priority-based processing

New in 8.0, the MTA supports the MT-PRIORITY SMTP extension defined in [RFC 6710 \(SMTP Extension for Message Transfer Priorities\)](#). An explicit MT-PRIORITY value overrides any

of the older Priority: header line base priority settings, or the MTA's old size-based priority adjustment effects. See the discussion of the `mt_priority` MTA option for a description of how MT-PRIORITY values are mapped to the older Priority: values. So a message with an explicit MT-PRIORITY value will get priority handling based on the mapping of that MT-PRIORITY value to the older Priority: value. Only a Sieve filter `setmtpriority action` can override a message's explicitly specified MT-PRIORITY value.

The standardized Priority: header field, defined in [RFC 2156 \(MIXER: Mapping between X.400 and RFC 822/MIME\)](#), is respected by the MTA. Note that this MTA message processing priority, as indicated in a Priority: value, affects MTA processing priority: that is, it affects when the MTA processes that message especially in contention with other messages of differing priority, and potentially how long the MTA continues to reattempt delivery of messages experiencing temporary delivery failures. Note that this Priority: effect is completely different from the sort of user e-mail client display feature requested via a Precedence: or Importance: header field (though users sometimes confuse and conflate these different sorts of effects). Keep in mind that time-criticality (processing priority) is not necessarily the same thing as importance: a relatively insignificant message may be time-critical due to time-limited relevance, or conversely a message of significant importance may concern an event far removed in time.

Priority: is an MTA-level feature, not typically appropriate for arbitrary users to set themselves (though specially privileged users such as administrators may desire and be entitled to access to priority adjustment), and may come at a "cost", whether that cost is simply additional work by the MTA, additional charges to the user, or an effect that delivery attempts abort sooner (thus causing messages to potentially be less likely to get through, being bounced sooner rather than getting additional delivery attempts if the message can't be delivered "quickly"). Importance: or Precedence: on the other hand, are user-level features, appropriate for users to set on the messages they send to request special handling by recipients or special display features when recipients' e-mail clients display the messages.

By default Priority: values are honored and make a "difference" in the MTA's message handling, affecting handling by the Job Controller, though that "difference" is generally small and hardly noticeable.

The Job Controller sorts messages, by priority, into separate internal processing queues. With just default configuration, the Job Controller will preferentially process "urgent" messages before "normal" messages before "non-urgent" messages for messages all eligible for delivery at the same time. However, under normal circumstances there are so many messages flowing through the MTA so quickly, with so many messages being handled in parallel, that this sort of difference in handling based on priority tends to be pretty much moot. Unless there's a big enough backlog of newly submitted messages that the usually fairly "immediate" delivery attempts are a bit delayed, the Job Controller's automatic sorting of messages by priority doesn't much matter; instead, with no backlog, each freshly submitted message can get an essentially "immediate" delivery attempt, regardless of the message's priority.

When it comes to delivery retries on messages that didn't get through on first attempt, there the default MTA configuration (with only the `backoff` channel option set) is to retry urgent messages at shorter periods than normal messages, and retry nonurgent messages at longer periods. This may be precisely controlled further via the `prioritybackoff` channel option settings. So by default, priority does have an (in principle) noticeable effect on the delivery retries scheduled by the Job Controller -- but once you're in the regime of having to retry to deliver, the messages are by definition "delayed", and the exact length of retry interval may matter little compared to the time scale of whatever is causing the delivery attempts to fail. So again, this is a difference, but perhaps not a difference that "matters" much.

As for how long messages are retained if they continue to fail to be deliverable, there the MTA default (the default handling by the return job) is that priority doesn't have an effect -- though you can change that via the [prioritynotices](#) channel option settings. But note that even if your own MTA hasn't been configured to make a distinction, other MTAs that your messages pass through in principle might care and bounce "urgent" messages sooner if they fail delivery---this is one of the potential "costs" of higher priority processing mentioned above.

Now, if you choose to configure different [Job Controller delivery execution windows](#), you can potentially have very different handling of different priorities of messages, even for freshly submitted new messages. With that (non-default) configuration, then you could see very noticeably different handling of different priorities of messages. See the [nonurgent_delivery](#), [normal_delivery](#), and [urgent_delivery](#) Job Controller options.

In particular, one type of use of priority delivery execution windows is to defer the processing of "non-urgent" messages to "off hours"; *e.g.*, in legacy configuration set in the `job_controller.cnf` file:

```
! Add to a pool definition to postpone delivery attempts of non-urgent
! messages to the hours between 11:00 PM and 4:00 AM any day, or any time
! Sunday.
!
NONURGENT_DELIVERY=23:00 - 04:00, Sun 00:00 - 23:59
```

Or in Unified Configuration, set via:

```
msconfig> set option job_controller.job_pool:SMTP_POOL.nonurgent_delivery "23:00 - 04:00, Sun 00:00 - 23:59"
```

When priority "matters", note that the MTA has ways of overriding the priority specified on a Priority: header line. The priority that the MTA actually uses is the *effective priority*---normally what is specified on a Priority: header line, unless overridden in some way. The [priorityblocklimit channel options](#), for instance, and the new in Messaging Server 7.4-0.01 system Sieve action `setpriority` can be used to override the original processing priority, setting a new effective processing priority. As of the 8.0 release, an explicitly specified MT-PRIORITY value on a message will override any Priority: header based setting, or MTA size-based priority adjustment. Only the Sieve filter [setmtpriority action](#) can override an explicitly specified MT-PRIORITY value.

42.2 Job Controller default configuration

The MTA is distributed with an initial Job Controller configuration that is a suitable starting point for most sites. The default configuration defines three pools: (1) one named DEFAULT with a job limit of ten, to be used for miscellaneous channels, (2) one named IMS_POOL with a job limit of two, to be used for running ims-ms channel jobs, and (3) one named SMTP_POOL with a job limit of ten, to be used for outbound TCP/IP SMTP/LMTP channel jobs.

The following figure shows a default configuration in Unified Configuration.

Sample Job Controller option settings in Unified Configuration

```
msconfig> show job_controller
role.job_controller.tcp_ports = 27442 (3)
role.job_controller.job_pool:DEFAULT.job_limit = 10 (5), (6)
role.job_controller.job_pool:IMS_POOL.job_limit = 2
```

```

role.job_controller.job_pool:SMTP_POOL.job_limit = 10
role.job_controller.channel_class:bitbucket.master_command = IMTA_BIN:bitbucket (7)
role.job_controller.channel_class:bsmtp*.master_command = IMTA_BIN:bsout_master (8)
role.job_controller.channel_class:bsmtp*.slave_command = IMTA_BIN:bsin_master (8)
role.job_controller.channel_class:conversion*.master_command = IMTA_BIN:conversion
role.job_controller.channel_class:defragment.master_command = IMTA_BIN:defragment
role.job_controller.channel_class:filter_discard (novalue) (11)
role.job_controller.channel_class:hold.master_command = IMTA_BIN:reprocess
role.job_controller.channel_class:ims-ms*.master_command = IMTA_BIN:ims_master (9)
role.job_controller.channel_class:ims-ms*.max_life_askwork = 20000 (9)
role.job_controller.channel_class:ims-ms*.max_life_time = 14400 (9)

role.job_controller.channel_class:native.master_command = IMTA_BIN:l_master
role.job_controller.channel_class:pipe*.master_command = IMTA_BIN:pipe_master
role.job_controller.channel_class:process*.master_command = IMTA_BIN:reprocess
role.job_controller.channel_class:reprocess*.master_command = IMTA_BIN:reprocess
role.job_controller.channel_class:sms*.master_command = IMTA_BIN:sms_master
role.job_controller.channel_class:tcp_*.master_command = IMTA_BIN:smtp_client (10)
role.job_controller.channel_class:uucp_*.master_command = IMTA_BIN:uucp_master
role.job_controller.channel_class:uucp_*.slave_command = IMTA_BIN:uucp_slave
instance.job_controller.secret (suppressed) (1)

```

In legacy configuration, the Job Controller configuration is stored in a file, `job_controller.cnf`. And in legacy configuration, this Job Controller configuration file is required. If it is not present or its contents are incorrect the Job Controller will not start.

There is no need to modify the Job Controller configuration settings (the Job Controller configuration file in legacy configuration), unless you choose to add pools, modify pool parameters, modify global Job Controller settings (such as debugging), or add processing information for locally developed channels.

In legacy configuration, if you do wish to make such modifications, you should not alter the Job Controller configuration file itself (since it will be replaced when you upgrade the MTA and in legacy configuration you will lose your modifications), but rather should create a `job_controller.site` file in the MTA table directory containing your own definitions. The Job Controller configuration file will read in this site supplied file, if it exists.

A sample Job Controller configuration file is shown below.

```

!
! Global defaults
!
SECRET=abc123 (1)
SLAVE_COMMAND=NULL (2)
TCP_PORT=27442 (3)
!
! Site specific pools and channels are read
! indirectly if this include file exists.
!
<IMTA_TABLE:job_controller.site (4)
!
! Pool definitions
!
[POOL=DEFAULT] (5)
JOB_LIMIT=10 (6)
!

```

Job Controller default configuration

```
[POOL=IMS_POOL]
JOB_LIMIT=2
!
[POOL=SMTP_POOL]
JOB_LIMIT=10
!
! Channel definitions
!
[CHANNEL=bitbucket]      (7)
MASTER_COMMAND=IMTA_BIN:bitbucket
!
[CHANNEL=bsmtp*]        (8)
MASTER_COMMAND=IMTA_BIN:bsout_master
SLAVE_COMMAND=IMTA_BIN:bsin_master
!
[CHANNEL=conversion*]
MASTER_COMMAND=IMTA_BIN:conversion
!
[CHANNEL=defragment]
MASTER_COMMAND=IMTA_BIN:defragment
!
[CHANNEL=ims-ms*]       (9)
MAX_LIFE_AGE=14400
MAX_LIFE_CONNS=20000
MASTER_COMMAND=IMTA_BIN:ims_master
!
[CHANNEL=native]
MASTER_COMMAND=IMTA_BIN:l_master
!
[CHANNEL=pipe*]
MASTER_COMMAND=IMTA_BIN:pipe_master
!
[CHANNEL=process*]
MASTER_COMMAND=IMTA_BIN:reprocess
!
[CHANNEL=sms*]
MASTER_COMMAND=IMTA_BIN:sms_master
!
[CHANNEL=tcp_*]         (10)
MASTER_COMMAND=IMTA_BIN:smtp_client
!
[CHANNEL=reprocess*]
MASTER_COMMAND=IMTA_BIN:reprocess
!
[CHANNEL=uucp_*]
MASTER_COMMAND=IMTA_BIN:uucp_master
SLAVE_COMMAND=IMTA_BIN:uucp_slave
!
[CHANNEL=hold]
MASTER_COMMAND=IMTA_BIN:reprocess
!
[CHANNEL=filter_discard] (11)
```

The key items in the above examples are:

1. This global option sets a "secret" used on this host by the Job Controller to verify its internal communications.
2. Set a default SLAVE_COMMAND for subsequent [CHANNEL] sections.
3. This global option defines the TCP port number on which the Job Controller listens for requests.
4. Attempt to include the optional, site-supplied `job_controller.site` file (in which sites may place their site-specific customizations, so as to retain such customizations after upgrading).
5. This [POOL] section defines a queue named "DEFAULT". This pool will be used by all channels which do not specify a pool name using the [pool channel option](#).
6. Set the JOB_LIMIT for this pool to 10.
7. This [CHANNEL] section applies to a channel named [bitbucket](#). The only definition required in this section is the MASTER_COMMAND which the Job Controller issues to run this channel. (Note that the `bitbucket` channel normally never needs to run, so normally this image is never executed, since in normal use messages supposedly "enqueued" to the `bitbucket` channel are instead merely deleted--however, a `bitbucket` channel image does exist to delete messages, and if a message does exist in the `bitbucket` channel queue, perhaps due to being manually placed there by the MTA administrator, then this channel image can "process" it ---that is, delete it.) Since no wildcard appears in the channel name, the channel name must match exactly.
8. This [CHANNEL] section applies to any channel whose name begins with [bsmtp_*](#). For this channel, both a MASTER_COMMAND and a SLAVE_COMMAND are necessary. Since this channel name includes a wildcard, it will match any channel whose name begins with "bsmtp_".
9. This [CHANNEL] section applies to any channel whose name begins with [ims-ms*](#). For this channel, used to deliver to the Messaging Server Message Store, it is a good idea to set the [max_life_time](#) (formerly MAX_LIFE_AGE) and [max_life_askwork](#) (formerly MAX_LIFE_CONNS) Job Controller (channel class) options to let the channel jobs "persist" (rather than being "recycled" in favor of a new channel job) for relatively extended periods.
10. This [CHANNEL] section applies to any channel whose name begins with [tcp_*](#); this includes SMTP over TCP/IP and LMTP over TCP/IP channels. This section only defines (the Job Controller only knows/cares about) a MASTER_COMMAND defining the SMTP/LMTP client "half" of any such channel; the slave "half" of any such channel (SMTP servers or LMTP servers) is handled by the [Dispatcher](#).
11. This [CHANNEL] section applies to the [filter_discard channel](#). The absence of any MASTER_COMMAND in this section is intentional.

42.3 Job Controller options

A number of options affect Job Controller operation.

In Unified Configuration, Job Controller options are set using the `msconfig` utility. The option names are set and inspected under `job_controller` for options affecting overall/global

Job Controller operation, or under `job_controller.job_pool:pool-name` for options affecting a specific, named pool, or under `job_controller.channel_class:channel-name-prefix` for options affecting a certain channel or type of channel; e.g.,

```
msconfig> show job_controller.tcp_ports
role.job_controller.tcp_ports = 27442
msconfig> show job_pool:DEFAULT.*
role.job_controller.job_pool:DEFAULT.job_limit = 10
msconfig> show channel_class:tcp*
role.job_controller.channel_class:tcp_*.master_command = IMTA_BIN:smtp_client
```

In legacy configuration mode, Job Controller options are set in the `job_controller.cnf` file.

Generally, the Job Controller must be [restarted](#) in order for its option changes to take effect. But as [restarting the Job Controller is undesirable, tends to degrade performance, and should be avoided in operation](#) except when *truly* necessary, the `imsimta cache -change` utility provides a means to change the effective values for certain, especially operationally relevant, Job Controller options "on the fly", without requiring a Job Controller restart.

See also the [logfile options](#) set as `job_controller.logfile.*`.

42.3.1 enable Option Under job controller

The `enable Job Controller option` is used to enable the [Job Controller](#) daemon. This defaults to the value of the `mta.enable` (Unified Configuration) or `local.imsimta.enable` (legacy configuration) option. When the `job_controller` is enabled, the default for `schedule.task:purge.enable` (Unified Configuration) or `local.schedule.purge.enable` (legacy configuration) is 1. In addition, the default for `schedule.task:return_job.enable` (Unified Configuration) or `local.schedule.return_job.enable` (legacy configuration) is also 1 in this case.

42.3.2 debug Option Under job controller

The `debug Job Controller` option sets a bit mask for various types of debugging. When debugging is enabled, it is written to the Job Controller log file. That file is located in the MTA log directory, `SERVERROOT/log/`, and named `job_controller.log-uniqueid` where *uniqueid* is a unique name disambiguifying the file name. (Note that the [imsimta purge utility](#) understand the *uniqueids* and can be used to purge back older log files.)

42.3.3 listenaddr Option Under job controller

The `listenaddr Job Controller` option (formerly `INTERFACE_ADDRESS` in legacy configuration) can be used to specify the IPv4 address interface to which the Job Controller should bind for listening for its own communications; (see the [tcp_ports](#) Job Controller option in Unified Configuration, which replaced the legacy configuration `TCP_PORT` Job Controller option). By default, the Job Controller binds to the [tcp_ports](#) (legacy configuration `TCP_PORT`) on all IP addresses. But for systems having multiple network interfaces each with its own IP address, it may be useful to listen only on a particular interface. Note that if `listenaddr` is specified, then that is the only interface IP address to which the Job Controller will bind for its internal use.

Note that the [interfaceaddress](#) channel option provides a capability for specifying which interface address a TCP/IP channel uses for outgoing connections and messages; this is quite separate from the Job Controller's internal use of interface address(es). Also note that the [Dispatcher](#) has its own setting for `listenaddr`, controlling which IP address a particular Dispatcher service listens on.

42.3.4 job_limit Option

The `job_limit` Job Controller `job_pool` option relates to pool configuration, rather than channel configuration: it is set either for a specific Job Controller `job_pool`, or if set directly under `job_controller` it becomes the default for all pools which don't explicitly specify a `job_limit`. The option's value specifies the maximum number of jobs that a pool can execute in parallel. Execution of a request will use a UNIX process, so this corresponds to the maximum number of UNIX processes you allow a pool to use. The `job_limit` applies to each pool individually; the maximum total number of jobs is the sum of the `job_limit` parameters for all pools.

Note that multithreaded processes, *e.g.*, the [TCP/IP client program](#) (`tcp_smtp_client`) and the Message Store delivery channel program (`ims-ms_master`), may run multiple threads within a single process, hence in that sense multiple "delivery jobs" within a process. The discussion of jobs here refers to *job processes*.

Setting `job_limit` to 0 effectively stops a pool: it can't process any requests.

See [Job Controller default configuration](#) for examples of `job_limit` values.

Channel configuration normally specifies a pool for each channel to run in via the `channel pool` option (or in legacy configuration via the `pool` keyword on each channel defined in the MTA configuration file, `imta.cnf`). By having different channels run in different pools, they can be kept from competing with each other. By grouping "related" channels to run in the same pool, they can share (compete) for resources (processing slots) amongst each other, while not being allowed to compete for processing slots with those channel(s) running in other pools. Use of the `maxjobs` keyword on a channel can limit how much of the `job_limit` of the pool that the channel runs in the channel is allowed to use; this is normally only of interest when multiple channels are running in the same pool, being used to limit how much of the pool's `job_limit` a particular channel is allowed to use.

In legacy configuration, `job_limit` is set either within a [POOL] section, or globally at the top of the `job_controller.cnf` file. If set outside of a section, it will be used as the default by any [POOL] section which doesn't specify `job_limit`. This option is ignored inside of a [CHANNEL] section.

Note that the `imsimta cache -change` utility may be used to change a channel's effective `job_limit` "on the fly".

42.3.5 master_command Option

The `master_command` Job Controller `channel_class` option relates to channel configuration, rather than pool configuration: it is set for a specific channel or type of channel under a `job_controller.channel_class:channel-name-prefix`. The option's value specifies the full path to the command to be executed by the UNIX process created by the Job Controller in order to run the channel and dequeue messages outbound on that channel. This option is not available for a `job_pool`.

Prior to Messaging Server 7.0.5, there were no defaults in the MTA code for `master_command` value(s); however, MTA initial configuration created a `job_controller.cnf` file which set an appropriate `master_command` value for each type of normally installed channel. As of Messaging Server 7.0.5, each normally installed channel class has an appropriate default `master_command` value. (These defaults are different for each sort of channel class.)

Note that in legacy configuration, the `master_command` option is specified inside `[CHANNEL=...]` sections of the Job Controller configuration file, or if specified at the top in the global section becomes a default for all channels which don't explicitly specify a `master_command`. This option is ignored inside of a `[POOL]` section.

42.3.6 max_cache_messages Option

The Job Controller keeps information about messages in an in-memory structure (the "queue cache database"). This is essentially a cached-in-memory index to the message files currently on disk. In the event that a large backlog of messages builds up on disk, the Job Controller may need to limit the size of this in-memory structure so as not to allow memory usage to grow excessively. If the number of messages in the backlog exceeds the Job Controller's currently computed maximum messages value (see below -- the Job Controller uses the specified `max_cache_messages` value as a starting point, but during operation may adjust that value up or down according to circumstances), then information about subsequent messages is not kept in the in-memory queue cache database. Mail messages are not lost because they are always written to disk, (and the disk queue area will get scanned by the Job Controller eventually, and remaining messages will then be detected and processed) but such messages 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 `imsimta cache -sync` command. The initial default for `max_cache_messages` is 100,000. But the Job Controller, while running, may adjust this size either up or down, depending on circumstances.

To manually adjust the effective `max_cache_messages` size "on the fly" (without having to restart the Job Controller to get a change to the `max_cache_messages` option to take effect), you may use the `imsimta cache -change` utility:

```
# imsimta cache -change global -max_messages=value
```

Note that exceeding the current maximum messages value means that subsequent messages can be expected to be processed "out-of-order". `max_cache_messages` is not truly a performance-related option -- increasing it will not improve message throughput nor will decreasing it (unless decreased to an absurdly low number) have much effect on message processing speed. Rather, its purpose is to place a limit on the Job Controller's memory usage, and its operational effect is to limit on how many messages the Job Controller will attempt to schedule in "first in, first out" order. The Job Controller's scheduling is normally roughly "first in, first out", as modified by `message processing priority`, response speed of destination being delivered to, and effects of "bunching up" messages to the "same" host onto the same process and even process thread. But once the current maximum messages value is exceeded, the Job Controller is only attempting "first in, first out" scheduling on the messages it has in its in-memory queue cache database; the remaining messages don't get a shot at being processed until later (once the Job Controller initiated channel jobs have managed to deliver a lot of the original backlog of message). And scanned messages picked up via a disk scan (whether manually executed `imsimta cache -sync` or the Job Controller's automatic rescans such as due to dropping sufficiently below the current maximum messages value), are all considered to be of lower processing priority than newly submitted, normal priority messages.

In legacy configuration mode, `maxmessages` was a deprecated synonym for `max_cache_messages`; that old `maxmessages` name is not a synonym in Unified Configuration, where it refers instead to a different option for the Message Store.

42.3.7 max_life_askwork Option

The `max_life_askwork` [Job Controller option](#) imposes another limit on channel master job life expectancy, in addition to the limit set for that channel type by the `max_life_time` Job Controller option setting. The `max_life_askwork` option sets a limit on the number of times a channel master job can ask the Job Controller if there are any messages for the job to process. If this option is not specified for a channel, then the global default value is used. If no default value is specified, 300 is used.

42.3.8 max_life_time Option Under job controller

The `max_life_time` [Job Controller option](#) requests that Job Controller channel master jobs only be kept around for the specified number of seconds. (See also the `max_life_askwork` Job Controller option which imposes another limit on channel master job persistence.) When the [Job Controller creates a channel master job process](#), a countdown timer is set to the specified number of seconds. When the countdown time has expired, the channel job is subject to being shut down. The `max_life_time` option may be set either directly under the `job_controller` group, as a default for all channel types, or may be set at the `channel_class` level to affect solely that type of channel. The Job Controller's default is 4 hours (14400).

The Job Controller's `max_life_time` option had the synonyms `max_life_age` and `max_age`, which are now deprecated.

42.3.9 notice_time Option

RESTRICTED: This option affects Job Controller internal processing, intended for future use.

The [Job Controller](#) occasionally performs some housekeeping intended for future use in generating [notification messages](#); (for current operation, see instead the `notices` channel option). The `notice_time` option controls when the Job Controller performs such housekeeping. It is of the form `HH:MM/hh:mm`, or `/hh:mm`. `hh:mm` is the interval in hours and minutes between operations, and `HH:MM` is a notional clock time at which the operation starts. If `HH:MM` is not specified, the first scan will be `hh:mm` after the Job Controller starts. If `HH:MM` is specified, the first scan will be the first time that `HH:MM + n * hh:mm` is greater than the Job Controller start time. The default is `/00:30`.

42.3.10 port Option Under job controller

The `port` Job Controller option specifies the TCP port that the Job Controller uses for inter-process communication.

42.3.11 rebuild_parallel_channels Option

On startup the [Job Controller](#) scans the queues for messages left over from a previous invocation. It reads from several channel queues at the same time to somewhat "balanced" (across queues) its initial message delivery attempts. The `rebuild_parallel_channels` Job Controller option limits the number of channel queues to be scanned simultaneously. The default is 12.

Note that an `imsimta cache -change -global -parallel_rebuild=n` command may be used to change the parallelism "on the fly".

42.3.12 secret Option Under job controller

The `secret` Job Controller option specifies a string secret used on this host by the Job Controller to verify its internal communications.

Note that the Job Controller `secret` option's value is normally set to a randomly generated string during initial configuration. It is safe to change this value; *however*, do note that the value normally is (and *should be*) set as an `instance` option, rather than a `role` option, using a different value on each host (a different value set for each instance).

42.3.13 slave_command Option

The `slave_command` Job Controller `channel_class` option relates to channel configuration, rather than [pool configuration](#); it is set either for a specific channel or type of channel under a `job_controller.channel_class:channel-name-prefix`. The option's value specifies the full path to the command to be executed by the UNIX process created by the Job Controller in order to run the channel and poll for any messages inbound on the channel. Note that many MTA channels do not have a `slave_command`.

```
msconfig> set role.job_controller.channel_class:bsmtp*.slave_command "IMTA_BIN:bsin_master"
```

Prior to Messaging Server 7.0.5, the default value of `slave_command` for all channel classes was the special value NULL. As of Messaging Server 7.0.5, the `bsmtp` channel class has a default value for `slave_command` (see above), while all other normally installed channels have as default the special value NULL.

Note that in legacy configuration, the `slave_command` option is specified inside `[CHANNEL=...]` sections of the Job Controller configuration file, or if specified at the top in the global section becomes a default for all channels which don't explicitly specify a `slave_command`. If a channel does not have a `slave_command` (has no sensible slave direction), the reserved value NULL should be specified. This option is ignored inside of a `[POOL]` section.

42.3.14 stressblackout Option

The Job Controller ignores repeated indications from channels saying they are stressed for a short time after a stressed indicator is received. The `stressblackout` Job Controller option specifies for how long (in seconds) to ignore repeated such indications. The default is 60 seconds, *i.e.*, one minute.

42.3.15 stresstime Option

The `stresstime` Job Controller option specifies for how long, in seconds, a channel remains at an elevated stress level after a stress notification is received. The default is 120 seconds, *i.e.*, two minutes.

42.3.16 stressfactor Option

When a channel's stress level is raised, the `threaddepth` for the channel is multiplied by the `stressfactor` Job Controller option value to obtain a temporarily increased effective `threaddepth`, thereby reducing the Job Controller's aggressiveness in spawning new threads and jobs for the channel. The default is 5.

42.3.17 `unstressfactor` Option

When a channel's stress level is lowered, the effective `threaddepth` for the channel is divided by the `unstressfactor` Job Controller option value, thereby increasing the Job Controller's aggressiveness in spawning new threads and jobs for the channel. If not explicitly set, the `stressfactor` value is taken as the default.

42.3.18 `stressjobs` Option

When a channel's stress level is raised, the effective job limit for the channel is decreased by the `stressjobs` Job Controller option value. The default is 2.

42.3.19 `unstressjobs` Option

When a channel's stress level is lowered, the effective job limit for the channel is increased by the `unstressjobs` Job Controller option value. The default is the same value set for `stressjobs`.

42.3.20 `synch_time` Option

The Job Controller occasionally scans the channel queue directories for message files it does not know about, to insert corresponding entries into its in-memory queue cache database. The `synch_time` option controls when. The value is of the form `HH:MM/hh:mm`, or `/hh:mm`. `hh:mm` is the interval in hours and minutes between scans, and `HH:MM` is a notional clock time at which this starts. If `HH:MM` is not specified, the first scan will be `hh:mm` after the Job Controller starts. If `HH:MM` is specified, the first scan will be the first time the `HH:MM + n * hh:mm` is greater than the Job Controller start time. The default is `/04:00`.

This automatic disk scan is akin to performing a manual `imsimta cache -sync` operation.

42.3.21 `tcp_ports` Option Under job controller

The `tcp_ports` Job Controller option, `job_controller.tcp_ports`, specifies the TCP port on which the Job Controller should listen for request packets; that is, it is the port on which the Job Controller listens for its internal protocol communications. You generally do not want to change this option unless the default conflicts with another TCP application on your system. This is a global Job Controller option, set directly under `job_controller`; it is not available under channel or pool groups. The default is 27442.

42.3.22 Job Controller job pool options: **`nonurgent_delivery` (execution-window string),** **`normal_delivery` (execution-window-string),** **`urgent_delivery` (execution-window-string)**

The `nonurgent_delivery`, `normal_delivery`, and `urgent_delivery` Job Controller `job_pool` options each set an "execution window" for messages of the respective effective processing priority. The default is that all messages are eligible for processing at all times.

An execution window consists of up to five, comma-separated time windows. Each time window is either a daily window (a window of time allowed every day) of the form:

hh:mm - hh:mm

or a weekly window (a window of time allowed per week) of either of the forms

ddd hh:mm - ddd hh:mm

or (with the ending day assumed to be the same as the beginning day)

ddd hh:mm - hh:mm

For instance, a time window

`18:00 - 22:00`

means between 6:00 PM and 10:00 PM each day.

`20:00 - 06:30`

means between 8:00 PM and 6:30 AM each night.

`Sat 06:15 - 15:30`

means each Saturday between 6:15 AM and 3:30 PM.

`Wed 12:00 - Fri 00:00`

means between noon Wednesday and midnight Thursday/Friday (the midnight dividing Thursday from Friday). And thus an execution window specifying that processing is allowed any night or all day on weekends could be

`22:00 - 05:30, Sat 00:00 - Sun 23:59`

Note that the MTA can be configured to modify messages' effective processing priority based on certain criteria such as message size, see the `*blocklimit` channel options, or via the MTA's non-standard Sieve extension "setpriority". As of the 8.0 release, an explicitly specified `MT-PRIORITY` value for a message will override the older Priority: header value or the MTA's size-based effective processing priority adjustments, and in particular take precedence for Job Controller delivery execution window purposes. Only the MTA's non-standard Sieve extension "setmtpriority" can override an explicitly specified `MT-PRIORITY` value.

42.3.23 use_nslog Option Under job controller

The `use_nslog` Job Controller option may be set to 1 to enable use of `nslog()` for Job Controller debug log files. The default is 0. When `use_nslog` has been enabled, see also the [logfile options](#) that may be set as `job_controller.logfile.*`. Note that the `loglevel` option is *not* supported for the [Job Controller](#) as its debug level is controlled instead by use of the `debug` Job Controller option, `job_controller.debug`.

42.3.24 job_pool

The `job_pool` group (under `job_controller`) is not a Job Controller option itself, but rather a grouping of [Job Controller options](#) defining a particular named Job Controller processing pool. For instance:

```
msconfig> set job_controller.job_pool:OFFHOURS_POOL.job_limit 3
msconfig# set job_pool:OFFHOURS_POOL.normal_delivery "14:00 - 12:00, Sat 00:00 - Sun 23:59"
msconfig# set job_pool:OFFHOURS_POOL.nonurgent_delivery "17:00 - 9:00, Sat 00:00 - Sun 23:59"
```

This defines a pool named `OFFHOURS_POOL` which: allows at most three simultaneous jobs, limits delivery attempts of "normal" priority messages to occur outside the hours of noon until 2:00 PM on weekdays, and limits delivery attempts of "non-urgent" priority messages to occur outside business hours Monday through Friday. That is, if any channels are assigned via the `pool` channel option to run in this new `OFFHOURS_POOL` Job Controller pool, any messages enqueued to such channels will get delivery attempts only at times as follows: "urgent" priority messages can get delivery attempts at any time, as usual; "normal" priority messages will not get delivery attempts around lunch time on weekdays, but can get delivery attempts at any other times; "non-urgent" priority messages can only get delivery attempts outside business hours, *i.e.*, on weekends, or after 5:00 PM or before 9:00 AM on weekdays.

See [Job Controller operation](#) for further discussion of how the Job Controller utilizes such processing pools, see [Job Controller default configuration](#) for several examples of processing pool definition, and see the `pool channel option` for further details on how each channel is assigned to some such processing pool.

The other type of grouping of Job Controller options is under the `channel_class` group, used to set parameters on Job Controller channel job initiation and execution.

42.3.25 channel_class

The `channel_class` group (under `job_controller`) is not a Job Controller option itself, but rather a grouping of [Job Controller options](#) defining a particular named Job Controller channel, or class of channels having a certain name pattern. For instance:

```
msconfig> show channel_class:ims-ms*
role.job_controller.channel_class:ims-ms*.master_command = IMTA_BIN:ims_master
role.job_controller.channel_class:ims-ms*.max_life_askwork = 20000
role.job_controller.channel_class:ims-ms*.max_life_time = 14400
```

This defines the class of channels whose names begin `ims-ms`. Such channels have a [master program](#) (named `ims_master`), and have `max_life_askwork` and `max_life_time` options set to force delivery jobs for such channels to "time out" after certain work and time limits, so that a new, fresh delivery job will be created (as necessary) by the Job Controller.

See [Job Controller operation](#) for further discussion of how the Job Controller tracks messages and initiates channel jobs that run channel programs, see [Job Controller default configuration](#)

for several examples of channel definitions, see [Available channels](#) for a list of the normal channels supplied with the MTA, and see the `imsimta cache -change` utility for how to inform an already running Job Controller process of a new `channel_class` "on the fly".

The other type of grouping of Job Controller options is under the `job_pool` group, used to set parameters on Job Controller processing pools.

42.4 Checking that the Job Controller is running

You may check that the [Job Controller](#) is running with the command `imsimta process`. On UNIX, you should see output similar to that shown below, perhaps with additional jobs present if your system is currently processing messages.

```
# imsimta process
  USER  PID S  VSZ  RSS   STIME      TIME COMMAND
mailsrv 12435 S 32936 9672 13:54:01    0:00 /opt/SUNWmsgsr/lib/job_controller
mailsrv 12433 S 32480 8936 13:54:01    0:00 /opt/SUNWmsgsr/lib/dispatcher
```

Normally, the Watcher is configured to check periodically that the Job Controller is running (and start a new Job Controller, if there is none present).

Chapter 43 Compiling the MTA configuration

TBD

Chapter 44 Mail filtering and access control

44.1 Access mapping tables	44-2
44.1.1 PORT_ACCESS mapping table	44-3
44.1.2 INTERNAL_IP mapping table	44-6
44.1.3 Recipient access mapping tables	44-7
44.1.4 FROM_ACCESS mapping table	44-13
44.1.5 When access mapping table controls are applied	44-15
44.2 Defending against denial of service attacks	44-17

A common goal is to outright reject messages from (or to) certain users at the system level, or to limit the number of throttle the rate at which messages are accepted, or to institute more complex restrictions of message traffic between certain users, or to allow users to set up filters on their own incoming messages (including rejecting messages based on contents of the message headers). The MTA has a number of facilities in such areas, including:

- system level mapping tables such as `SEND_ACCESS`, `FROM_ACCESS`, and `MAIL_ACCESS` that permit both simple and sophisticated restrictions of message traffic based on source and destination and envelope From and To addresses---see [Access mapping tables](#);
- the `PORT_ACCESS` mapping table that permits restriction of SMTP connection attempts based on source IP address---see [Connection access control](#); if using the MMP as an SMTP proxy, see also the [MMP's access filters](#);
- user level (and system level) message filtering using Sieve filters, including sophisticated filtering based on message headers---see [Sieve filters](#).
- the general [MeterMaid](#) facility that can be used to count or track numbers of messages (or other "events" of interest) across processes and either perform "throttling" itself, or be queried from system level mapping tables or Sieve filters that then make access decisions based upon the MeterMaid counts.

Related topics discussed elsewhere include: different authentication mechanisms for different sorts of connections -- see [Controlling authentication mechanisms](#); and techniques falling under the general category of defending against denial of service attacks -- see [Defending against denial of service attacks](#).

Use of the `PORT_ACCESS` mapping table for connections to the MTA [Dispatcher](#) (*e.g.*, connections to the SMTP server) or [TCP wrappers](#) for client connections to the Message Store servers is a very efficient approach when rejection decisions can be taken based purely upon source IP address---see [Connection access control](#). Use of mapping tables such as `SEND_ACCESS`, `MAIL_ACCESS`, `FROM_ACCESS`, *etc.*, is an efficient approach when "envelope level" controls are desired---see [Access mapping tables](#). When users wish to implement their own personalized controls, or when message header and body content-based filtering is desired, the more general mail filtering approach using Sieve is likely appropriate---see [Sieve filters](#).

The MTA also uses mapping tables to check other sorts of access, including:

- Deciding which IP addresses are "internal": [INTERNAL_IP](#)
- Permitting use of specific SMTP commands:

- ETRN commands: [ETRN_ACCESS](#)
- BURL commands: [BURL_ACCESS](#)
- STARTTLS commands: [TLS_ACCESS](#)
- Controlling mailing list posting access:
 - [GROUP_AUTH](#)
 - Many [alias options](#) (or in legacy configuration, [alias file named parameters](#)) that name site-specific mapping tables, including alias options [alias_auth_mapping](#), [alias_cant_mapping](#), [alias_hold_mapping](#), [alias_nohold_mapping](#), [alias_moderator_mapping](#), [alias_sasl_auth_mapping](#), [alias_sasl_cant_mapping](#), and [alias_sasl_moderator_mapping](#)
- Controlling outbound SMTP connections and authentication:
 - [AUTH_ACCESS](#)
 - [IP_ACCESS](#)

44.1 Access mapping tables

There are several MTA [mapping tables](#) that may be used to control who may or may not connect to the SMTP/SMTP SUBMIT/LMTP servers, what destination hosts and IP addresses may be sent to, what connections may use certain SMTP commands, who may send mail, who may receive mail, or control who may post to mailing lists. For general information on the format and usage of MTA mapping tables, see [Mapping table format](#).

The [PORT_ACCESS mapping table](#) is used by the [Dispatcher](#) to control blocking of connections from particular IP addresses or IP address ranges, and to control use of different authentication mechanisms for different sorts of connections. The [PORT_ACCESS](#) mapping table in particular is relevant for certain techniques falling under the general category of see [defending against denial of service attacks](#). Although the [PORT_ACCESS](#) mapping table does not have access to message address information and hence does not permit the fine level granularity of, for instance, the [ORIG_MAIL_ACCESS](#) mapping table, and although it only applies to incoming SMTP/SMTP SUBMIT/LMTP over TCP/IP messages, note that for what it does do it is a very efficient approach (more efficient than using one of the later, address-based access mapping tables) since it rejects a disallowed host's connection attempt at the TCP level, before the channel dialogue (the SMTP/SMTP SUBMIT/LMTP transaction) has even begun.

The [FROM_ACCESS](#) mapping table is probed at the point of attempted message submission where the envelope From address has been provided; in SMTP terms, at the stage of the MAIL FROM: command. In particular, this is after the [PORT_ACCESS probe](#) (that decides whether to allow an SMTP/SMTP SUBMIT/LMTP connection) but before the [recipient address mapping tables probes](#) (that decide whether to allow particular recipient addresses). Another feature of the [FROM_ACCESS](#) mapping table is that it also has access to the authenticated sender information (SMTP AUTH information in particular).

The four [recipient access mapping tables](#), [ORIG_SEND_ACCESS](#), [SEND_ACCESS](#), [ORIG_MAIL_ACCESS](#), and [MAIL_ACCESS](#), can make use of envelope address information (as well as, in some cases, all the IP information available to the [PORT_ACCESS](#) mapping table). The nature of these mapping tables is very general, and allows per channel granularity, that is, channel-specific controls.

Of the [recipient access control mapping tables](#) applied at the SMTP RCPT TO command stage, the [MAIL_ACCESS](#) and [ORIG_MAIL_ACCESS](#) mapping tables 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. But when IP address information is not relevant to the desired controls, then use of `SEND_ACCESS` or `ORIG_SEND_ACCESS` may be simpler. And for some purposes, combining use of two or more of these tables may be convenient; see [When access mapping table controls are applied](#) for a discussion of the timing and ordering of when access mapping table controls are applied.

The [AUTH_REWRITE mapping table](#) is checked after the SMTP DATA is received, so that it has access not only to SMTP envelope fields but also to the header fields in the message itself. Thus while it is not usually the primary tool for checking sender access, as its check occurs later and thus is less efficient for outright blocking certain undesired senders, it can be very useful for enforcing site policy requirements regarding use of proper (perhaps authenticated) From: addresses, by rejecting messages that do not conform to policy.

44.1.1 PORT_ACCESS mapping table

The MTA [Dispatcher](#) is able to selectively accept or reject incoming connections to the services it manages such as SMTP, SMTP SUBMIT, or LMTP, 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 transport information in the form:

```
TCP | server-address | server-port | client-address | client-port
```

and try 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.

If no entry matched, then and only then will any [dns_verify_domain](#) lookups, as specified via a Dispatcher option (in particular in legacy configuration mode, in the Dispatcher configuration file), be performed, and the result of such a lookup is another way a connection may be refused. In particular, note that either an explicit rejection in `PORT_ACCESS`, or (as JES MS 6.0) a match without a rejection hence an "accept" effect will prevent [dns_verify_domain](#) lookups from occurring; this allows `PORT_ACCESS` to do initial filtering on connections, either "black listing" or "white listing" them, with [dns_verify_domain](#) taking effect only on any other source IP addresses.

The flag `$<` followed by an optional string causes the MTA to send the string to syslog (UNIX) if the mapping probe matches; the flag `$>` followed by an optional string causes the MTA to send the string to [syslog](#) (UNIX) if access is rejected.

If bit 1 (value 2) of the [log_connection MTA option](#) is set and the `$N` flag is set so that the connection is rejected, then also specifying the `$T` flag will cause a "T" entry to be written to the [MTA connection log](#). Note that running processes do not notice the periodic "roll-over" of the `mail.log_current` file into the `mail.log_yesterday` file, and the creation of a new `mail.log_current` file; such changes are normally noticed merely because new processes come into existence and see the "new" file. But in the case of the Dispatcher, which is normally a very long running process (normally not restarted except at times of certain configuration changes), this means that the Dispatcher, which is writing the "T" records, will continue writing to the "old" log file. For instance, after a `mail.log_current` file has been renamed to `mail.log_yesterday`, the Dispatcher will keep writing its "T" records to the file it "knew" about, now named `mail.log_yesterday`; it will not know to start writing

to `mail.log_current` unless and until the Dispatcher is restarted. (As of JES MS 6.2, the Dispatcher periodically (namely once an hour) forces a close and re-open of the connection log file.) So, if you are using "T" records, you may wish to restart your Dispatcher daily (at the time of log file rollover)---especially if you are running a version prior to JES MS 6.2.

The `PORT_ACCESS` mapping table, in addition to its normal use by the [Dispatcher](#), is also optionally probed again by the [SMTP server](#) and [LMTP server](#) for the purpose of determining the appropriate SASL rule set (when SMTP AUTH has been used during message submission); as of 7.0, the SMTP server probe of the `PORT_ACCESS` mapping table is unconditional (always performed). (However, the LMTP server probe of `PORT_ACCESS` is still conditional.) Or enabling bit 4 of the `log_connection` MTA option also causes the SMTP server and LMTP server to probe the `PORT_ACCESS` mapping table; in this case site-supplied text may be provided in the `PORT_ACCESS` entry to include in the SMTP server's and LMTP server's *application-info* field---a field which is used in certain types of log entries (such as "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. New in JES MS 6.3-0.15, such SMTP server probes of `PORT_ACCESS` will respect the `$N` (in the case of SMTP AUTH usage), `$>`, and `$<` flags, whereas in prior versions the SMTP server probe results were only relevant for setting the SASL ruleset and the optional logging text; as of the fix for 12208860 (Sun 6590888) (JES MS 6.3-5.02), `$N` rejections will be respected in all cases. Thus new in JES MS 6.3-0.15 for the special case of SMTP AUTH use, and true in general subsequently, the SMTP server probes of `PORT_ACCESS` can be used to achieve connection rejections (in this case performed by the SMTP server processes, rather than by the main Dispatcher process); for "simple" rejections it is more efficient to perform such rejections from the main Dispatcher process, but for potentially complex or "slow" rejections (such as rejections determined by the results of DNS verification lookups), deferring the rejection until the individual SMTP server process stage can avoid "bottlenecking" the main Dispatcher process waiting for a result of a probe. (LMTP server probes of `PORT_ACCESS` remain, as previously, relevant only for setting the SASL ruleset and the optional logging text.)

Table 44.1 `PORT_ACCESS` mapping flags

Flag	Description
<code>\$U</code>	(New in 6.3-0.15) Enable channel debugging . As of 7.3-11.01, this includes consulting the <code>mm_debug</code> and <code>os_debug</code> MTA options and enabling any debugging they specify. This is only supported for SMTP server probes of the <code>PORT_ACCESS</code> mapping table; it is not supported for Dispatcher probes of the <code>PORT_ACCESS</code> mapping table.
<code>\$/</code>	(New in 7.0-0.04) Set the "fast disconnect" flag for sessions that have not yet succeeded in starting a transaction; for such sessions, any subsequent disconnect is done with <code>SO_LINGER</code> enabled and a timeout of 0, which may clear slots quicker on intermediate firewalls and proxies.
<code>\$V</code>	(New in 7.0-0.04) Enable the MTA's private SMTP extensions XADR, XCIR, XGEN, and XSTA, overriding any SMTP server <code>DISABLE_*</code> TCP/IP-channel-specific options. This is only supported for SMTP server probes of the <code>PORT_ACCESS</code> mapping table; it is not supported for Dispatcher probes of the <code>PORT_ACCESS</code> mapping table.
<code>\$Y</code>	Allow access.
<code>\$T</code>	If bit 1 of the <code>log_connection</code> MTA option is set, and if a connection is rejected (<code>\$N</code> is also specified), then write a connection log file "T" record , including any of the optional <code>text</code> s specified

	with \$N. This is only supported for Dispatcher probes of the PORT_ACCESS mapping table; it is not supported for SMTP server or LMTP server probes of the PORT_ACCESS table.
	Flags with arguments, in argument reading order
\$Astring	(New in Messaging Server 7.4-18.01.) Set the HULA debug flags specified by the argument string; comma-separated flags can be "perf", "connect", "authserv", and "hula"; see the AUTH_DEBUG TCP/IP-channel-specific option . This is only supported for SMTP server probes of the PORT_ACCESS mapping table; it is not supported for Dispatcher probes of the PORT_ACCESS mapping table.
\$<string	Send string to syslog (UNIX) if probe matches.
\$>string	Send string to syslog (UNIX) if access is rejected.
\$Nstring	Reject access with the optional error text string.
\$Fstring	Synonym for \$Nstring, <i>i.e.</i> , reject access with the optional error text string.
text	If bit 4 of the log_connection MTA option is set, then the optional text text (which must appear subsequent to two vertical bar characters) may be included in the connection log "C" entry tm . This is only supported for SMTP server and LMTP server probes of the PORT_ACCESS mapping table; it is not supported for Dispatcher probes of the PORT_ACCESS mapping table.
\$Ddelay	(New in 6.3-0.15) Delay the banner flush by the specified number of centiseconds, overriding the BANNER_PURGE_DELAY value. This is only supported for SMTP server probes of the PORT_ACCESS mapping table; not supported for Dispatcher probes of the PORT_ACCESS mapping table.
\$Schannel-name	(New in Messaging Server 7.0-0.04) Set the specified channel as the source channel for this SMTP session. This is only supported for SMTP server probes of the PORT_ACCESS mapping table; not supported for Dispatcher probes of the PORT_ACCESS mapping table.
Flag comparisons	Description
\$:A	Match only when the probe is performed by the Dispatcher
\$;A	Match only when the Dispatcher is not performing the probe
\$:S	Match only when the probe is performed by an SMTP server or LMTP server
\$;S	Match only when neither an SMTP server nor an LMTP server is performing the probe

To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

Note that prior to JES MS 6.3-0.15, Dispatcher probes of the PORT_ACCESS mapping table could not make use of [LDAP callouts \(\\$...\[callouts\)](#).

The following example PORT_ACCESS 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|*|*                  $N500$ Bzzzzzzzzt$ thank$ you$ for$ playing.
```

Note that you will need to restart the Dispatcher---or as of JES MS 6.3-0.15 use the [imsmta reload utility](#) to reload the changed mappings file into running processes such as the Dispatcher---after making any changes to the PORT_ACCESS mapping table so that the Dispatcher will see the changes. (And if you're using a compiled MTA configuration, you'll 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 number based rejections; for more general control at the email address level, the [e-mail address access mappings](#) such as [SEND_ACCESS](#) or [MAIL_ACCESS](#) may be more appropriate.

44.1.2 INTERNAL_IP mapping table

Modern MTA configurations typically make use of an INTERNAL_IP [mapping table](#) as a convenient, single location for storing a site's list of "internal" IP addresses. MTA components that need to check whether or not a source IP address is "internal" then can make use of the INTERNAL_IP mapping table for this determination, rather than each component having its own separate list. So while knowledge and use of the INTERNAL_IP mapping table is not hard-coded into the MTA, it is a common configuration feature.

Typical component users of an INTERNAL_IP mapping table include: the [PORT_ACCESS mapping table](#) (to determine SASL ruleset), and a backwards-pointing [IP literal rewrite rule](#) (for the purpose of [switchchannel](#) "switching" to the [tcp_intranet channel](#)).

A typical INTERNAL_IP mapping table might appear something like the following, shown from within msconfig:

```
msconfig> show mapping:INTERNAL_IP.*
role.mapping:INTERNAL_IP.rule = host's-public-IP-address $Y
role.mapping:INTERNAL_IP.rule = $<192.168.0.0/16> $Y
role.mapping:INTERNAL_IP.rule = ${:::1} $Y
role.mapping:INTERNAL_IP.rule = 127.0.0.1 $Y
role.mapping:INTERNAL_IP.rule = * $N
```

or in legacy configuration:

```
INTERNAL_IP

host's-public-IP-address      $Y
$<192.168.0.0/16>             $Y
${:::1}                       $Y
127.0.0.1                     $Y
*                              $N
```

This sample INTERNAL_IP mapping table's rules explicitly match the host's public IP address, use a [subnet match](#) to match private IP addresses, and match IPv6 and IPv4 forms of loopback address. A final, fall-through wildcard \$N rule ensures that any IP addresses not listed/

matched earlier in the `INTERNAL_IP` table will fail the mapping check (not be considered "internal").

44.1.3 Recipient access mapping tables

The `ORIG_SEND_ACCESS`, `SEND_ACCESS`, `ORIG_MAIL_ACCESS`, and `MAIL_ACCESS` mapping tables may be used to control who may or may not 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; in addition, the `ORIG_MAIL_ACCESS` and `MAIL_ACCESS` mapping tables also have access to all the information available to the [PORT_ACCESS mapping table](#). Note that when the envelope To addresses are irrelevant and only the envelope From address matters, then use of the `FROM_ACCESS` mapping table, described in [FROM_ACCESS mapping table](#), may be more convenient and efficient.

If an `ORIG_SEND_ACCESS` or `SEND_ACCESS` mapping table exists, then by default for each recipient of every message passing through the MTA, the MTA will probe the `ORIG_SEND_ACCESS` and/or `SEND_ACCESS` mapping tables with a probe string of the form (note the use of the vertical bar character, |)

```
src-channel | from-address | dst-channel | to-address
```

where `src-channel` is the channel originating the message (*i.e.*, enqueueing/submitting the message); `from-address` is the address of the message's originator; `dst-channel` is the channel to which the message will be enqueued/submitted; and `to-address` is the address to which the message is addressed. Use of an asterisk in any of these fields causes that field to match any channel or address, as appropriate; see [Mapping table format](#) for additional matching characters and sequences.

Note that the addresses here are envelope addresses, that is, envelope From address and envelope To address. The `from-address` used is by default the so-called *mapped return address*: the envelope From address after simple address reversal has been applied to it (but not including, for instance, destination-specific address reversal). However, the MTA options `use_orig_return`, and (new in MS 6.3) `use_canonical_return`, and (new in MS 7.0) `use_auth_return`, can be used to probe with a different form of the envelope From 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.

The MTA options `access_orcpt`, `access_counts`, `include_conversiontag`, and `include_spare1` (replaced in MS 8.0 by `include_spare1`) can cause inclusion of additional fields in the probes. If the relevant bits of all these options are set (the options take bit-encoded integer arguments with distinct bits controlling the inclusion of the fields in the distinct mapping tables), then the format of probes with all optional fields enabled becomes

```
src-channel | from-address | dst-channel | to-address | orcpt | access-counts | list-of-tag
```

The optional `orcpt` field is the SMTP ORCPT field; that is, it will contain the address type (typically "rfc822") followed by a semicolon, followed by the "original" to address, *e.g.*, "rfc822;user@domain.com". The optional `access-counts` field contains a count of which recipient address (RCPT TO) this probe is for, followed by a forward slash, following by a count of the number of valid recipient addresses resulting from previous recipient address submissions (addresses resulting from previous RCPT TO commands), followed by a forward slash, potentially followed by additional counts expected to be added in future versions; so note that good practice is to always use an asterisk here, so that future additions will not cause

old entries to stop working. The optional `list-of-tags` field contains a comma-separated list of [conversion tags](#) already present on this message. (Note that conversion tags apply to entire message copies, different recipient conversion tags causing message copy "split up". But these recipient address probes occur before final recipient determination and hence before application of recipient conversion tags occurs.) The optional `s*` fields contain the values of the LDAP attributes named by the `ldap_spare_*` MTA options.

The `MAIL_ACCESS` mapping table is a superset of the `SEND_ACCESS` and `PORT_ACCESS` mapping tables; that is, it combines both the channel and address information of `SEND_ACCESS`, with the IP address and port number information of `PORT_ACCESS`. (See [PORT_ACCESS mapping table](#) for discussion of the `PORT_ACCESS` mapping table.) Similarly, the `ORIG_MAIL_ACCESS` mapping table is a superset of the `ORIG_SEND_ACCESS` and `PORT_ACCESS` mapping tables. The format for the default probe string for `MAIL_ACCESS` is

```
port_access-probe-info | app-info | submit-type | send_access-probe-info
```

and similarly the format for the default 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 (see [PORT_ACCESS mapping table](#)) in the case of incoming SMTP messages (including originally-incoming-SMTP messages that have been deferred to the [reprocess channel](#)), or will be blank otherwise, and `app-info` will usually be `SMTP/claimed-HELO-name` in the case of messages submitted via SMTP or `SMTP/claimed-HELO-name/TLS-crypto-info` in the case of messages submitted via SMTP over TLS (including the case of originally-incoming-SMTP messages that have been deferred to the `reprocess channel`), and blank otherwise. `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. And for the `MAIL_ACCESS` mapping table, `send_access-probe-info` consists of all the information usually included in a `SEND_ACCESS` mapping table probe (including, if the MTA options [access_orcpt](#), [access_counts](#), [include_conversiontag](#), and/or [include_spare](#)s are enabled, their respective additional fields). 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 (including, if the MTA options [access_orcpt](#), [access_counts](#), [include_conversiontag](#), and/or [include_spare](#)s are enabled, their respective additional fields). (See above for additional discussion of `SEND_ACCESS` and `ORIG_SEND_ACCESS` probes, as well as the [access_orcpt](#), [access_counts](#), [include_conversiontag](#), [include_spare](#)s MTA options.)

New in MS 7.0, the MTA option [mapping_paranoia](#) can cause vertical bars present within a field to be replaced by a different character, thereby ensuring that the only vertical bar characters in a probe are the field delimiter occurrences.

Now, if the probe string matches a pattern (*i.e.*, 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 recipient (envelope To) address is permitted.

If the mapping 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.+ (Unless an at-sign character, `@`, is part of the explicit rejection text, the MTA will append a colon and the recipient address to the end of the explicitly specified rejection text.) If no string is output (other than the `$N`, `$n`, `$F`, or `$f` flag), then some default rejection text will be used.

As of 6.2-0.01, the exact default text depends upon the value of the [access_errors](#) option; by default, if that option is 0, then the full SMTP error, including text, is:

```
550 5.7.1 unknown host or domain: recipient-address
```

This default rejection text is rather intentionally misleading for reasons of security, as under some circumstances it may be desirable to avoid revealing the real reason for a rejection. When no such security concerns apply, it is often more user-friendly to take the option of specifying some explanatory rejection text, either your own choice of text, or set [access_errors=1](#). If [access_errors](#) is 1, then the full SMTP error, including text, is: but if that option is 1, then the text is:

```
550 5.7.1 you are not allowed to use this address
```

Technical note: When the MTA generates a probe, it may also set various input flags specific to certain probe conditions: *e.g.*, the `A` input flag is set if SMTP AUTH has been used. Though these input flags are not part of the string part of the probe, they may be detected (tested) in a mapping entry via flag comparison tests in the entry template (right hand side). See the "Input flag comparisons" section of [Address access mapping table flags](#) for a full list. Input flags are separate from output flags: even in an iterative mapping, a flag set as output on one line will not become an input flag for possible subsequent probes, as it is the original input flags which are the input flags for all the probes. Sophisticated uses of the `*_ACCESS` mapping tables may make use of such input flag tests to more precisely specify under which circumstances to apply mapping effects.

See [Address access mapping table flags](#) for descriptions of additional flags. Note that input flags (set by the MTA prior to probing) are the only flags that may be tested using the input flag comparisons. Output flags, those set in the template (right hand side) of a mapping entry, are separate and not subject to such testing.

Table 44.2 Address access mapping table flags¹

Output flag	Description
<code>\$/</code>	(New in MS 7.0) Set the "fast disconnect" flag for sessions that have not yet succeeded in starting a transaction; for such sessions, any subsequent disconnect is done with <code>SO_LINGER</code> enabled and a timeout of 0, which may clear slots quicker on intermediate firewalls and proxies. ³
<code>\$!</code>	Disable (Sieve requested) vacation messages regarding this message ²
<code>\$*</code>	(New in MS 7.0 for <code>FROM_ACCESS</code> ; new in MS 7.0u2 for the other address <code>*_ACCESS</code> mappings) If used with <code>\$N</code> , force disconnect of the SMTP session.
<code>\$(+L</code>	(New in MS 7.0.5) If used with <code>\$M</code> in <code>FROM_ACCESS</code> , cause the captured message copy to be in journal format
<code>\$B</code>	Redirect the message to the bitbucket
<code>\$H</code>	Hold the message as a .HELD file
<code>\$J</code>	(New in MS 7.0u4) If used with <code>\$M</code> , cause generation of envelope "journal" format rather than the default DSN encapsulated format for the "capture" message copy ⁵

\$O	Forces single-copy-per-recipient message copy "split up", as if the single channel option were set on the relevant destination channel(s).
\$P	Force to enqueue to the reprocess channel . (For instance, this might be useful as part of <code>\$.text</code> handling when attempting an LDAP lookup that encountered an LDAP directory temporary problem; that is, in case of an LDAP lookup problem, defer to the reprocess channel.)
\$V	Force Sieve filter discard behavior for <i>all</i> recipients of this message copy.
\$v	(New in MS 7.0u3) Force Sieve filter discard behavior for solely this recipient ⁵ . For <code>FROM_ACCESS</code> , \$v remains equivalent to \$V as meaning force Sieve filter discard for all recipients.
\$Y	Allow access.
\$Z	Force Sieve filter jettison behavior (non-overridable discard) for <i>all</i> recipients of this message copy.
\$z	(New in MS 7.0u3) Force Sieve filter jettison behavior (non-overridable discard) for solely this recipient ⁵ . For <code>FROM_ACCESS</code> , \$z remains synonymous with \$Z as meaning force Sieve filter jettison for all recipients.
Output flags with arguments, in argument reading order ²	
\$Un	Enable channel (slave) debugging ; if the optional n argument is specified, it also sets the specified value for enqueue debugging .
\$~channel-name	(New in MS 7.0, as well as MS 6.3p1) Change source channel to <code>channel-name</code> -- this may be of especial interest for the case of incoming notification messages (incoming messages with empty envelope From). Note that when this feature is used and it actually <i>changes</i> the source channel, then the <code>FROM_ACCESS</code> check process is restarted; also, \$~ can only be applied once. ³
\$Jaddress	Replace original envelope From address with specified address ³
\$Kaddress	Replace original Sender: address with specified address ³ (Note that while \$K sets the MTA's internal value for the sender, hence will affect various access checks, whether or not a new Sender: header line should be added to the message displaying the newly specified sender address is controlled by the authrewrite channel option; that is, use of \$K does not, <i>on its own</i> , cause addition of a Sender: header line showing the new sender address.)
\$Iuser identifier	Check specified user for specified groupid (UNIX) and if not in the group, reject access (effectively sets the \$N output flag).
\$<string	Send <code>string</code> to syslog (UNIX) if probe matches; see also the sndopr_priority MTA options ⁴
\$>string	Send <code>string</code> to syslog (UNIX) if access is rejected; see also the sndopr_priority MTA option ⁴
\$Ddelay	Delay response for an interval of <code>delay</code> 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 recipient address access mapping tables).
\$Tprobe-tag	Prefix subsequent address *_ACCESS mapping table probes with the tag probe-tag. New in MS 7.0.5, the last such tag set may optionally be prepended to the AUTH_REWRITE mapping table probe.
\$Aheader	Add the header line header to the message; (see the spamfilter*_includeheaders MTA options if it is desired to have this added header line be visible to spam/virus filter packages).
\$Gconv-tag	(New in MS 6.0) Add the conversion tag (or comma-separated tags) conv-tag to the message.
\$Maddress	Capture a copy of the message , sending the captured and encapsulated copy to address. By default, this captured copy is DSN-encapsulated -- but see the no-argument, non-FROM_ACCESS \$J output flag above.
\$Sx\$Sx,y\$Sx,y,z	FROM_ACCESS ⁶ : Set an effective blocklimit , and optionally recipientlimit , and optionally recipientcutoff for the transaction. Prior to MS 6.3, these values were minimized with whatever other such limits were already in effect; as of MS 6.3 these values override any global MTA option or source-based such limits that may already be effect (but destination-based limits will still be applied, later). ³
\$,x	(New in 6.1) Set a spam level value x (between -10000 and 10000). If the message already had a spam level, this is a " spamadjust " effect (adds or subtracts the specified amount x from the prior spam level). Note that such a spam level/spamadjust effect applies to all recipients (even for the recipient-specific mapping tables such as SEND_ACCESS) in order for tests to see if one of the recipients is a "honeypot" address.
\$Qlanguage\$Qlanguage	(New in MS 6.3) Set a preferred language and optionally a preferred country ³
\$(postmaster-address	FROM_ACCESS ⁶ : (New in MS 6.3) Set an override postmaster address ³
\$(postmaster-address	(New in MS 6.3) Set an override postmaster address if none was previously set ³
\$+Ename value	(New in MS 7.0u2) Set the Sieve environment item name to value
\$+Rn string	(New in MS 7.0u2) FROM_ACCESS ⁶ : Opt-in to spam/virus filtering . n specifies which spam/virus filter package; string is the opt-in string to pass to that spam/virus filter package. ³
\$Xerror-code	Issue the specified error-code extended SMTP error code if rejecting the message; if the first digit of the extended SMTP error code x.y.z is a 4, then the rejection will be issued as a temporary rejection, 452 4.y.z, instead of the usual 550 5.y.z sort of permanent rejection
\$Nstring	Reject access with the optional error text string

\$Fstring	Synonym for \$Nstring, <i>i.e.</i> , reject access with the optional error text string
\$(file-spec)	[ORIG_][SEND MAIL]_ACCESS ⁶ : If rejecting a message to a mailing list (or other envelope-From-overridden sort of message) with other than a 4xx (temporary) error, override the usual <code>error_text_*</code> error text option values with values from the file <code>file-spec</code> ; the values should be specified one per line in the file, in the order of the <code>error_text_*</code> options as shown in <code>error_text_*</code> MTA options. ⁵
\$Surl	[ORIG_][SEND MAIL]_ACCESS ⁶ : Apply the Sieve filter obtained from resolving <code>url</code> ⁵
+\$Rn string	(New in 7.0u2) [ORIG_][SEND MAIL]_ACCESS ⁶ : Opt-in to spam filtering. Only applied if neither \$N nor \$F is set. <code>n</code> is which spam/virus filter package; <code>string</code> is the opt-in string to pass to that spam/virus filter package. ⁵
Input flag comparisons	Description
\$()	(New in MS 7.0) Match only if external material (<i>e.g.</i> , an envelope address) in the probe contained a vertical bar
\$(;)	(New in MS 7.0) Match only if no vertical bars were present in any external material in the probe
\$(A)	Match only if SMTP AUTH (authenticated submission) has been used
\$(;A)	Match only if SMTP AUTH (authenticated submission) has <i>not</i> been used
\$(D)	Match only if DELAY delivery receipts have been requested (<i>e.g.</i> , NOTIFY=DELAY) ⁵
\$(;D)	Match only if DELAY delivery receipts have <i>not</i> been requested ⁵
\$(E)	(New in MS 6.3) Match only if ESMTP has been used
\$(;E)	(New in MS 6.3) Match only if ESMTP has not been used
\$(F)	Match only if FAILURE delivery receipts have been requested (<i>e.g.</i> , NOTIFY=FAILURE) ⁵
\$(;F)	Match only if FAILURE delivery receipts have <i>not</i> been requested ⁵
\$(L)	(New in MS 6.3) Match only if LMTP has been used
\$(;L)	(New in MS 6.3) Match only if LMTP has not been used
\$(P)	(New in MS 7.0) Match only if POP-before-SMTP was used
\$(;P)	(New in MS 7.0) Match only if POP-before-SMTP was not used
\$(R)	(New in MS 8.0) Match if the current, enqueueing channel is an "internal" channel such as the reprocess channel
\$(;R)	(New in MS 8.0) Match if the current, enqueueing channel is something other than an "internal" channel
\$(S)	Match only if SUCCESS delivery receipts have been requested (<i>e.g.</i> , NOTIFY=SUCCESS) ⁵
\$(;S)	Match only if SUCCESS delivery receipts have <i>not</i> been requested ⁵
\$(T)	Match only if TLS has been used

\$;T	Match only if TLS has <i>not</i> been used
\$:V	(New in MS 7.0u1) Match only if recipient address expanded via an alias ⁵
;\$;V	(New in MS 7.0u1) Match only if recipient address did not expand via an alias ⁵

¹ These flags are relevant for the SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, ORIG_MAIL_ACCESS, and FROM_ACCESS mapping table, except where footnotes indicate special restrictions. Note that the PORT_ACCESS mapping table supports a somewhat different set of flags.

² To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

³ Available for the FROM_ACCESS mapping table only

⁴ 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.

⁵ Not available for the FROM_ACCESS mapping table.

⁶ This output flag has a different meaning and/or occurs in a different position, relative to other output flags, for FROM_ACCESS vs. the other address *_ACCESS mapping tables. See the rest of this table for the other occurrence of this flag, describing its operation for the other type of mapping table.

+ Note that it is up to whatever is attempting to send the message whether the MTA rejection error text is actually presented to the user who attempted to send the message. In particular, in the case when 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.

44.1.4 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 the MTA will probe the table with a probe 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 (including originally-incoming-SMTP messages that have been deferred to the [reprocess channel](#)), or will be blank otherwise, and app-info will usually be SMTPclaimed-HELO-name in the case of messages submitted via SMTP, or SMTP/claimed-HELO-name/TLS-crypto-info in the case of messages submitted via SMTP over TLS (including the case of originally-incoming-SMTP

messages that have been deferred to the `reprocess` channel), and blank otherwise. `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 (*i.e.*, queueing the message); `from-address` is the address of the message's purported originator (that is, the envelope From address); note that as of Messaging Server 7.0u2, the MTA options `use_orig_return`, and (new in 6.3) `use_canonical_return`, and (new in Messaging Server 7.0) `use_auth_return`, can be used to probe with a different form of the envelope From address. `auth-from` is the authenticated originator address, if such information is available (*e.g.*, from an SMTP AUTH command, in which case it is the user's `mail` attribute value which is returned from the SASL library code and substituted in this field of the probe), or blank if no authenticated information is available.

The `include_conversiontag` and `include_spare` (or `include_spare1` as of MS 8.0) MTA options can cause inclusion of additional fields in the `FROM_ACCESS` probes. If the relevant bits of these options are set, then the format of the probes with all optional fields enabled becomes

```
port_access-probe-info|app-info|submit-type|src-channel|from-address|auth-from|list-of-tags|s1|s2|s3|s4|s5
```

The optional `list-of-tags` field contains a comma-separated list of [conversion tags](#) already present on this message. (Note that conversion tags apply to entire message copies, different recipient conversion tags causing message copy "split up". But these recipient address probes occur before final recipient determination and hence before application of recipient conversion tags occurs.) The optional `s*` fields contain the values of the LDAP attributes named by the `ldap_spare_*` MTA options.

Now, if the probe string matches a pattern (*i.e.*, 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 from that particular From: address is permitted. If the mapping output contains any of the flags `$N`, `$n`, `$F`, or `$f`, then the enqueue from 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.¹ If no string is output (other than the `$N`, `$n`, `$F`, or `$f` flag), then default rejection text will be used,

550 5.7.1 Initial access check failure

See [Address access mapping table flags](#) for descriptions of additional flags.

As of JES MS 6.2p4, the initial configuration of the MTA generates a [minimal FROM_ACCESS mapping table that disables generation of vacation messages back to typical list "owner" addresses](#).

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 `$J` flag, or to set a different "sender" address via the `$K` flag. Although the envelope From and sender address are not always seen in message headers, they can be quite significant for various functional operations (such as access checks). And depending upon other configuration, they may potentially end up visible in header lines (envelope From in the Return-path: header line added during final delivery, and sender address in a Sender: header line if added due to use of the `authrewrite` channel option). For instance, this mapping table can be used to cause the original envelope From address to simply be replaced by the authenticated address, when an authenticated address is present:

```
FROM_ACCESS
```

```
* |SMTP* |* |tcp_local |* |      $Y
* |SMTP* |* |tcp_local |* |*     $Y$J$4
```

¹ Note that it is up to whatever is attempting to send the message whether the MTA rejection error text is actually presented to the user who attempted to send the message. In particular, in the case when FROM_ACCESS is used to reject an incoming SMTP message at the MAIL FROM: stage of the SMTP dialogue, 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.

44.1.4.1 Initial FROM_ACCESS mapping table

Initial configuration of the MTA normally generates a basic FROM_ACCESS mapping table that uses the \$! flag to [disable generation of vacation messages](#) back to envelope From addresses that typically correspond to list owner addresses.

```
msconfig> show mapping:FROM_ACCESS.*
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |MAILER-DAEMON@* |* $!$Y
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |LISTSERVE*@ |* $!$Y
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |majordomo@* |* $!$Y
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |*-request@* |* $!$Y
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |*-owner@* |* $!$Y
role.mapping:FROM_ACCESS.rule = * |SMTP* |* |* |owner-*@* |* $!$Y
```

This corresponds to legacy configuration of:

```
FROM_ACCESS
```

```
! Entries to block certain submissions normally would be inserted here,
! above the intended-to-be-final entries that while permitting submission,
! merely disable any potential "vacation" effect.
!
! The following entries disable Sieve "vacation" action on lists sorts
! of addresses, as recommended by the Sieve "vacation" extension draft.
!
* |SMTP* |* |* |MAILER-DAEMON@* |*      $!$Y
* |SMTP* |* |* |LISTSERVE*@ |*        $!$Y
* |SMTP* |* |* |majordomo@* |*        $!$Y
* |SMTP* |* |* |*-request@* |*        $!$Y
* |SMTP* |* |* |*-owner@* |*          $!$Y
* |SMTP* |* |* |owner-*@* |*          $!$Y
```

44.1.5 When access mapping table controls are applied

The MTA checks access control mapping tables as early as possible. Exactly when this happens depends upon the e-mail protocol in use---when the information that must be checked becomes available.

The Dispatcher consults the [PORT_ACCESS mapping table](#) to decide whether or not to accept the connection -- before even accepting a connection. In the case of an accepted SMTP connection handed over to an SMTP server process thread, then (as of Messaging Server 7.0) each SMTP server process thread makes its own additional probe of `PORT_ACCESS` after the hand off of a connection from the Dispatcher, prior to issuing the server's SMTP banner line. (If the "client" is the MMP and it sends an XPEHLO command, that resets the effective source IP address and so the SMTP server process thread will then do yet another `PORT_ACCESS` probe.) There are trade-offs in performance between the Dispatcher check and the server process check of `PORT_ACCESS`; see the [PORT_ACCESS mapping table](#) discussion for additional details.

Continuing with the SMTP protocol, where addresses are presented in the initial part of the attempted message handover, well before the message data itself would be handed over, note that a [FROM_ACCESS](#) rejection will occur in response to the MAIL FROM: command, *before* the sending side ever gets to send the recipient information let alone the message data, while a [recipient access mapping table](#) (`ORIG_SEND_ACCESS`, `SEND_ACCESS`, `ORIG_MAIL_ACCESS`, or `MAIL_ACCESS`) sort of rejection will occur in response to the RCPT TO: command, *before* the sending side ever gets to send the message data. Thus if an SMTP message is rejected due to such a `*_ACCESS` mapping table rejection, the MTA never even accepts or sees the message data, thereby minimizing the overhead of performing such rejections.

In contrast, note that an [AUTH_REWRITE mapping table](#) which (as of JES MS 6.2-0.01) may reject messages, or a Sieve script "refuse" message rejection, since they in principle may make use of information from the message as a whole (message header lines and body content), cannot be applied until after the message data has been received by the MTA. In particular, in the case of an SMTP message, these forms of rejection cannot occur until after the end of the DATA phase of processing, thus they involve more overhead (both network traffic, and MTA processing overhead) than rejections performed via a `*_ACCESS` mapping table.

So the order in which access mapping tables are checked is:

- [PORT_ACCESS](#) (by Dispatcher),
- [PORT_ACCESS](#) (by SMTP server as of Messaging Server 7.0),
- [TLS_ACCESS](#) (by SMTP server after successful STARTTLS negotiation), or [ETRN_ACCESS](#) (by SMTP server when an ETRN command is received),
- [FROM_ACCESS](#) (after SMTP MAIL FROM),
- [ORIG_SEND_ACCESS](#), [SEND_ACCESS](#), [ORIG_MAIL_ACCESS](#), and [MAIL_ACCESS](#) (after SMTP RCPT TO, after the MTA's address rewriting and alias processing has been applied, all four are checked in the order listed),
- [BURL_ACCESS](#) (by SMTP SUBMIT server when a BURL command is received instead of regular message DATA),
- [AUTH_REWRITE](#) (after SMTP DATA).

In the list above, failing a check at one stage normally aborts message acceptance and processing and hence no further checks need be performed. But as regards the recipient address based mapping tables `ORIG_SEND_ACCESS`, `SEND_ACCESS`, `ORIG_MAIL_ACCESS`, and `MAIL_ACCESS`, note that they are all checked at the same stage of processing (which is after address rewriting and alias processing); if multiple of these recipient address access

control mapping tables exist, then the MTA will check them all (and any side-effects they cause will take place) in the order just listed (which is a point to keep in mind when using such mapping tables to obtain side-effects such as addition of header lines, logging to `syslog`, *etc.*).

44.2 Defending against denial of service attacks

A denial of service attack is where an attacker tries (intentionally or inadvertently) to overwhelm your system by flooding you with e-mail.

In some cases, adding a simple static mapping entry to unconditionally reject messages from the problem address or site is a sufficient defense, particularly if you know ahead of time (or can quickly detect) that the attack is occurring; see [Access mapping tables](#) and [Client access to Message Store servers](#). In other cases, however, you may either wish to automate dynamic detection of message volume upswings sufficient to be considered an attack. Or you may not wish to reject *all* messages from the problem address or site and instead wish merely to "turn down the volume", *i.e.*, slow down the flow to a level more easily managed by your users or system. For instance, you may be under a practical or legal mandate to accept certain messages, or good messages may be mixed in with the bad message flow; in such a case turning down the volume to a manageable level allows the good messages a chance to get into your system while preventing the bad messages from overloading your resources.

The `PORT_ACCESS` and `SEND_ACCESS` mapping tables ---as well as related mapping tables discussed in [Access mapping tables](#)---can be used in more sophisticated ways than simple unconditional entries to achieve such goals, and can indeed be hooked into dynamic, heuristic routines to decide "Yea" or "Nay" on accepting messages, such as [MeterMaid](#), or [dns_verify routines](#), or calling out to PureMessage IP Blocker via the [pmxb1 routine](#), or calling out to a third-party anti-spam/anti-virus package that supports so-called "early verdicts" via the [mm_check_reputation routine](#), or callouts to site-provided routines.

First, on the most simple level, the `PORT_ACCESS`, `SEND_ACCESS` or related mapping tables can take a random argument, effectively having the MTA "flip a coin" each time it needs to decide whether to accept a connection or message, respectively, or in the case of `SEND_ACCESS` and related mapping tables whether to sideline a message; see the [\\$? substitution](#) (under Mapping tables) for details.

For more sophisticated needs, the `PORT_ACCESS`, `SEND_ACCESS`, or related mapping tables can call out to facilities such as [MeterMaid](#), or the [dns_verify routines](#), or call out to site-supplied shareable image routines; see [Site-supplied routine substitutions](#) (under Mapping tables) for details. Such routines can, if you wish, use PMDF API calls to access [MTA counters](#) information; this can allow for heuristic decisions based on recent message load, comparing MTA counters levels at one sampled time with MTA counters levels when checked a little later; *e.g.*, "lots of messages came in to the `tcp_local` channel in the last few minutes, so let us reject additional connection attempts for the moment" or whatever decision basis you decide to implement.

As of Messaging Server 7.0u2, use of a [LOG_ACTION mapping table](#) calling out to [MeterMaid](#) may be an even better approach for making dynamic decisions.

If you have a site approach for monitoring syslog notices, note that the [log_messages_syslog](#) and [log_connections_syslog](#) MTA options can be enabled to cause message and connection entries, respectively, to also be sent to syslog.

As of the 8.0 release, the [Sieve "memcache" operator](#) and [Sieve "metermaid" operator](#) are available to query and update memcache or [MeterMaid](#), respectively, from within a Sieve script. Sieve script powerful logic can thereby allow for especially flexible or "targetted" updates of memcache or MeterMaid tables, if desired, when such tables are to be queried for connection blocking or throttling type purposes.

The heuristics for making dynamic decisions about accepting or rejecting messages tend to be very site specific, and involve a variety of critical issues. Note also that sites may wish to keep the details of their own heuristic algorithms secure. Sites interested in implementing their own denial of service prevention techniques may wish to obtain specialized consulting assistance.

Particularly when implementing dynamic rejection mechanisms, the following [TCP/IP-channel-specific options](#) may be of interest:

- [ALLOW_TRANSACTIONS_PER_SESSION](#), [ALLOW_RECIPIENTS_PER_TRANSACTION](#), and [TRANSACTION_LIMIT_RCPT_TO](#). The [ALLOW_TRANSACTIONS_PER_SESSION](#) option can be used to limit the number of messages accepted during a particular connection. After refusing a number of connection attempts from a particular site, once you do let them connect, they are liable to have a backlog of messages for your site which they will try to deliver during that connection. If you are attempting to "slow down" how much mail you accept from that site, you likely will want to use this option to say, in effect, "enough for now" after some point in the connection. Similarly, the [ALLOW_RECIPIENTS_PER_TRANSACTION](#) option can be used to limit the number of recipients allowed for a particular message submission (additional recipients during that submission attempt being rejected with a temporary error, so that they may be tried again later); this can be useful in slowing down the number of recipients handed over during a single message transaction. Thus both these options may be useful protecting against a denial of service attack in the form of a rapid flood of messages blanketing large numbers of your users. The [TRANSACTION_LIMIT_RCPT_TO](#) option modifies the effect of [ALLOW_RECIPIENTS_PER_TRANSACTION](#), in that it controls at which SMTP command (RCPT TO: or MAIL FROM:) the "extra" recipient addresses are rejected.
- [REJECT_RECIPIENTS_PER_TRANSACTION](#). The [REJECT_RECIPIENTS_PER_TRANSACTION](#) option can be considered a more aggressive version of [ALLOW_RECIPIENTS_PER_TRANSACTION](#); whereas [ALLOW_RECIPIENTS_PER_TRANSACTION](#) allows up to the specified number of recipients to be accepted (while additional recipients get a temporary rejection) thereby merely "slowing down" the incoming messages, [REJECT_RECIPIENTS_PER_TRANSACTION](#) causes the MTA to issue a temporary rejection (after the DATA command) for *all* recipients of a message if too many recipients are attempted. That is, with [REJECT_RECIPIENTS_PER_TRANSACTION](#), a message will not be allowed in at all until the sending side decreases the number of recipients it is trying to submit during a single transaction. So for a sufficiently flexible sender, [REJECT_RECIPIENTS_PER_TRANSACTION](#) is simply a somewhat more forceful way of slowing down the message flow. But for a sender that is not so flexible about retrying sending a message with fewer recipients per transaction, [REJECT_RECIPIENTS_PER_TRANSACTION](#) may result in the message not getting through at all.
- [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#). The [ALLOW_REJECTIONS_BEFORE_DEFERRAL](#) option causes the MTA, after the specified number of recipients have failed (been determined to be invalid addresses), to reject (with a temporary error) all further recipients in that transaction, good or bad. That is, this option penalizes message submissions that include a lot of bad recipient addresses (on the theory that such message submissions may be cases of dictionary-based or automatically-generated lists of recipients).

- [SIZE_DELAY_THRESHHOLDS](#), [SIZE_DELAY_AMOUNTS](#), [RECIPIENT_DELAY_THRESHHOLDS](#), [RECIPIENT_DELAY_AMOUNTS](#), [TRANSACTION_DELAY_THRESHHOLDS](#), [TRANSACTION_DELAY_AMOUNTS](#). These options are specifically available to progressively "slow down" the acceptance of incoming messages once specified thresholds have been exceeded.

The above items focus on MTA features. However, many sites will also have a spam/virus filter package that may be capable of maintaining heuristic data concerning connections; if such a spam/virus filter package has useful such abilities and is configured to report or make available relevant such data to the MTA, as for instance via a [mm_check_reputation routine call from the PORT_ACCESS mapping table](#), that can be yet another useful tool in the arsenal for defending against denial of service attacks.

Chapter 45 Spam and virus filtering

45.1 Brightmail spamfilterN_config_file	45-2
45.2 ClamAV spamfilterN_config_file	45-4
45.3 ICAP spamfilterN_config_file	45-5
45.4 Milter spamfilterN_config_file	45-5
45.5 SpamAssassin spamfilterN_config_file	45-7
45.6 Archive spamfilterN_config_file	45-9
45.7 Spamfilter early verdicts	45-10
45.8 Milter implementation	45-11
45.8.1 MILTER_MACROS mapping table	45-14
45.8.2 Milter single recipient extension	45-15
45.9 imexpire invoking spamfilter packages	45-16

The MTA supports calling out to up to eight spam/virus filter packages while processing incoming messages. The configuration of how the MTA should locate and communicate with such spam/virus filter packages is controlled via [spamfilter MTA options](#); the configuration of which routes of messages traffic receive which spam/virus filtering may be controlled via [spamfilter channel options](#); in addition to, or instead of, channel-level control, the MTA also supports per address (either sender or receiver) opt-in to spam/virus filtering via various [user-level](#) or [domain-level](#) LDAP attributes, and via [alias_optinN](#) alias options.

Generally, spam/virus filter package verdicts regarding specific messages are interpreted by the MTA as requesting correspondingly configured [Sieve filter actions](#). Via the interaction of various types and levels of Sieve filters in the MTA's [Sieve hierarchy](#), complex logic weighting or comparing verdicts from multiple spam/virus filter packages can be achieved. The [Sieve spamtest](#) and [virustest extensions](#) can be particularly useful in the context of consulting multiple spam/virus filter packages.

With those spam/virus filter packages that support [connection-based "early verdicts"](#), the MTA can also make use of such information as a special application of its general [Mailing filtering and access control](#) functionality. See also the topics of [Defending against denial of service attacks](#) and [Triggering effects from transaction logging with LOG_ACTION](#).

Note that many spam/virus filter packages do not themselves support returning different verdicts for different recipients of a single message. However, [Brightmail](#) does; and as of Messaging Server 7.0.5.33, Oracle's [milter implementation](#) supports a [single recipient extension](#) to effect different milter actions for different recipients. Furthermore, note that even with a spam/virus filter package that does not itself support returning different verdicts for different recipients, different per-recipient eventual effects can be achieved (at the cost of some additional configuration complexity) in a couple of ways. One way to achieve different effects for different users even with a spam/virus filter package that does not support per-recipient verdicts is to have the spam/virus filter package result "mark up" a message in one way or another (*e.g.*, by adding a header line, or [setting a spam level](#)) and then make use of the MTA's [Sieve hierarchy](#) to take individualized actions in user-level Sieve filters. Another way to achieve individualized effects even with a spam/virus filter package that has no such inbuilt support is to assign different MTA spamfilter "slots" (that is, different N in the [spamfilterN_*](#) MTA options) to different configurations of a single spam/virus filter package, and then "opt in" users to the appropriate "package" configurations, either at the domain level or at the individual user level; see the [ldap_domain_attr_optinN](#) and [ldap_optinN](#) MTA options.

Integrating spam/virus filter package use as a "call-out" via [spamfilter MTA options](#) is generally much preferable in functionality to use of an "in front" or "on the side" separate, dedicated spam/virus filter SMTP host, or even site or third-party written filtering MTA channel, for reasons including:

- the call-out approach permits "up front" address validation by the MTA, thus avoiding wasting time on invalid (perhaps dictionary attack generated) recipient addresses;
- the call-out approach preserves SMTP responsibility and full SMTP features (*e.g.*, message size limits, message notification handling requests, message authentication data, *etc.*) for incoming messages with the MTA whose primary function that is;
- the call-out approach tends to maximize performance and throughput of messages; in particular, it avoids the overhead of additional full transfers of the messages and avoids SMTP performance bottlenecks or message buildups on third-party SMTP implementations.

However, the MTA does also support hooking in site or third-party written MTA channels via the [alternate conversion channel](#) approach.

The Message Store's `imexpire` utility also supports operating in a mode where it calls out to spam/virus filter packages "as if" it were an MTA channel, applying a subset of Sieve filter actions corresponding to the spam/virus filter package verdicts again "as if" it were an MTA channel. This feature thus permits [post-delivery scanning of messages for spam/virus](#), and removal from the Message Store of such messages determined (post-delivery) to be spam or virus-contaminated.

45.1 Brightmail spamfilterN_config_file

When using Brightmail (and running it on a separate, remote system), a [spamfilterN_config_file](#) MTA option might be set as

```
msconfig> set mta.spamfilter1_config_file /opt/mailwail/config.remote
```

For Brightmail, the file that this option names may contain an extensive list of option settings. Listed below in [Some Brightmail configuration options](#) are some especially relevant valid options, but for a full list of valid options, consult Brightmail documentation. Brightmail configuration file option names are not case-sensitive. However, the values (right hand sides) are potentially case-sensitive. The Brightmail configuration file uses LDIF format; in particular, options are specified as

```
option-name: option-value
```

Table 45.1 Some Brightmail configuration file options

Option name	Meaning	Typical setting for MTA use
<code>blSWPrecedence</code>	A message may receive multiple forms of processing hence multiple verdicts; for instance, receive verdicts of both <code>spam</code> and <code>virus</code> . This option specifies the order in which verdicts are processed for messages with multiple verdicts. Verdicts should be separated with the hyphen character, <code>-</code> . A verdict appearing first (left-most) in the hyphen-separated list will be processed before a verdict appearing next (to the right), <i>etc.</i> . Supported verdicts are <code>virus</code> and <code>spam</code> . Brightmail recommends that <code>virus</code> verdicts be processed first.	<code>blSWPrecedence: virus-spam</code>

bISWClientDestinationDefault	This option specifies how to handle messages that have no verdict (for instance, that are not gray or virus and hence do not require additional filtering). Usually such messages should simply be delivered normally, that is, to the user's usual inbox. The value <code>inbox</code> has this meaning, and hence is usually the recommended setting.	<code>bISWClientDestinationDefault: inbox</code>
bISWLocalDomain	Occurrences of this option specify which domain(s) are considered local, hence which domains get the handling specified by <code>bISWClientDestinationLocal</code> options (as opposed to handling specified by <code>bISWClientDestinationForeign</code> options).	<code>bISWLocalDomain: your-domain-1.com</code> <code>bISWLocalDomain: your-domain-2.com</code> ... <code>bISWLocalDomain: your-domain-n.com</code>
bISWClientDestinationLocal	This option specifies <i>verdict-name destination-data</i> pairs for recipients in a local domain (a domain specified by a <code>bISWLocalDomain</code> option). (<code>bISWClientDestinationForeign</code> options control the handling for recipients in any other domains). That is, a setting of this option controls what verdict string (destination-data) is passed to the MTA corresponding to a particular Brightmail verdict. The MTA is normally configured (via a <code>spamfilterN_null_action=data,discard;</code> MTA option setting) to consider a null <i>destination-data</i> value to be a request to discard the message. And since the MTA's default action, given a non-null verdict (non-null destination-data), is to file to a folder of the same name as the verdict (destination-data), due to the default: <code>spamfilterN_string_action=data:,require "fileinto"; fileinto "\$U";</code> This typically but not necessarily amounts to <i>verdict-name folder-name</i> pairs, with a null (empty) folder-name meaning to discard the message. But more precisely, it truly consists of <i>Brightmail-verdict-name MTA-verdict-string</i> pairs, where the <i>MTA-verdict-string</i> can be specified in an MTA option of the form <code>spamfilterN_verdict_M=MTA-verdict-string</code> to correspond to an action specified in a <code>spamfilterN_action_m</code> option, and with a null <i>MTA-verdict-string</i> being handled in accordance with the MTA's corresponding <code>spamfilterN_null_action</code> option. Supported <i>verdict-name</i> values include <code>spam</code> and <code>virus</code> .	The default settings are: <code>bISWClientDestinationLocal: spam junkmail</code> <code>bISWClientDestinationLocal: virus </code> which by default has the effect that messages that Brightmail considers to be virus will be discarded while messages that Brightmail considers to be spam will be filed to a recipient's "junkmail" folder.
bISWClientDestinationForeign	Analogous in syntax and meaning to <code>bISWClientDestinationLocal</code> settings above, but applying to recipients who are not in one of the <code>bISWLocalDomain</code> domains.	
bISWClientOptin	This option specifies the handling of opt-in decisions by the Brightmail Client, and for use with the MTA must be set to <code>TRUE</code> .	<code>bISWClientOptin: TRUE</code>
blswcServerAddress	This option specifies the IP address(es) and port(s) of one or more Brightmail servers. The syntax is <i>ip:port</i> , <i>ip:port,...</i>	<code>blswcServerAddress: ip:port</code> or <code>blswcServerAddress: ip-1:port-1, ip-2:port-2, ..., ip-n:port-n</code>
bISWCommonDebugFilename	This option specifies the location of Brightmail's own debug log file. It can be set to any of <code>syslog</code>	<code>blswcCommonDebugFileName:</code>

	or <code>syslog syslog-facility</code> (the default when logging to <code>syslog</code> if the facility is not set is <code>mail</code>), <code>stderr</code> , or a <i>path-to-filename</i> .	<code>/opt/mailwall/logs/common_log</code>
<code>blCommonDebugLevel</code>	This option enables Brightmail's own debugging. The value is a comma-separated list of pairs of values. The first subvalue in each pair is either a positive integer (specifying a section of Brightmail code - use reserved for Brightmail) or the keyword <code>ALL</code> , meaning that all Brightmail code is debugged. The second value in each pair is an integer between 0 and 7, specifying the severity of the error as defined in syslog . Brightmail recommends that for problem situations, this option be set to <code>ALL, 6</code> . And Brightmail recommends that after solving the difficulty, that the option be set back to <code>ALL, 4</code> .	<code>blCommonDebugLevel: ALL, 4</code>
<code>blswcDebugFileName</code>		<code>blswcDebugFileName: /opt/mailwall/logs/bmclient_log</code>
<code>blswcDebugLevel</code>		
<code>blswsDebugFileName</code>		<code>blswsDebugFileName: /opt/mailwall/logs/bmsserver_log</code>
<code>blswsDebugLevel</code>		

Material in this table is taken from the *Brightmail Solution Suite 4.0 for iPlanet Messaging Server* manual. While presented again here for convenience, that manual should be considered the definitive source for Brightmail documentation.

45.2 ClamAV spamfilterN_config_file

When using ClamAV, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter3_config_file /opt/SUNWmsgsr/config/clamav.dat
```

With ClamAV, the file that this option names may contain the following options:

- `DEBUG` (integer; default is 0).
- `USE_INSTREAM` (0 or 1; default is 0). New in Messaging Server 7.0u3. The default 0 selects that ClamAV "STREAM" request is sent; setting 1 selects that ClamAV "INSTREAM" request is sent.
- `MESSAGE_BUFFER_SIZE` (integer; default is 1048576).
- `HOST` (hostname or IP address). A value must be specified for this option; (its presence is required).
- `PORT` (integer). A value must be specified for this option; (its presence is required).
- `SOCKS_HOST` (string).
- `SOCKS_PORT` (integer; default is 1080).
- `SOCKS_USERNAME` (string).
- `SOCKS_PASSWORD` (string).
- `TIMEOUT` (integer; default is 3600).

- MODE (integer; default is 0).
- FIELD (string; default is "Virus-Test").
- VERDICT (string).

45.3 ICAP spamfilterN_config_file

When using ICAP, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter4_config_file /opt/SUNWmsgsr/config/icap.dat
```

With ICAP, the file that this option names may contain the following options:

- DEBUG (integer; default 0)
- TIMEOUT (integer; default 3600)
- SOCKS_HOST (string)
- SOCKS_PORT (integer; default 1800)
- SOCKS_USERNAME (string)
- HOST (string or IP address); this option is required (a value must be set)
- PORT (integer; default 1344)
- MODE (integer; default 0)
- FIELD (string; default "Virus-Test")
- VERDICT (string)

45.4 Militer spamfilterN_config_file

When using Militer, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter4_config_file /opt/SUNWmsgsr/config/milter.dat
```

With milter (supported as of JES MS 6.3), the file that this option names may contain the following options:

- CONNECT_TIMEOUT (integer). (New in 8.0) If this option is not set, it defaults to the value set for the TIMEOUT option. A non-positive setting for CONNECT_TIMEOUT, whether explicitly set or inherited from TIMEOUT, will result in using a CONNECT_TIMEOUT value of 60.
- CONTEXT_EDITS (integer; default is 1). (New in 8.0.1) The milter interface expresses header modification actions in terms of offsets, e.g., "delete the third occurrence of the Authentication-results: header field" or "replace the value of the first occurrence of the DKIM-Signature: field with ...". For the most part these actions have obvious analogues in Sieve using the index extension. However, when multiple milters acting in parallel modify

the same header field it's possible for the changes to overlap and produce anomalous results. This can be ameliorated by converting offsets into references to the header field's value, something the Sieve editheader extension also supports. The `CONTEXT_EDITS` option controls whether or not milter header modification actions are translated from offsets into value references. A non-zero value (the default) enables this translation; a zero value disables it.

- `DEBUG` (integer; default is 0). Non-zero values enable increasingly higher levels of debug output: a value of 1 enables basic debugging; a value of 2 enables, for instance, hex dumps of the milter responses; a value of 3 is also meaningful, enabling output of the octets of the milter response and as of the 8.0 release, additional other debug output such as debugging of use of the [MILTER_MACROS mapping table](#).
- `DEFER_MESSAGE_TRANSFER` (integer; default is 0). Normally messages are transferred to the milter server as they are presented to the MTA. Setting `>DEFER_MESSAGE_TRANSFER` to a non-zero value defers the transfer until after the preceding spamfilter plugin has completed its actions, at which point the message header and body are transferred to the milter server from the MTA's internal storage areas. Normally this option is used in conjunction with setting the `IMMEDIATE_HEADER_MODIFICATIONS` option on a previous milter spamfilter plugin, which results in the modifications made by the previous milter being visible to the current milter.
- `IMMEDIATE_HEADER_MODIFICATIONS` (integer, default 0). By default the milter interface converts milter header modification actions to Sieve actions. Setting this option to a non-zero value will cause the plugin to modify the MTA's internal copy of the message header directly; no Sieve actions will be generated. IMPORTANT NOTE: This option should ONLY be used with plugins enabled on the basis of the source channel; use with plugins enabled via destination channels will cause inconsistent results.
- `RESETDEBUG` (integer; default is 0). Setting `RESETDEBUG` enables milter debugging conditionally: only if channel debugging enabled. (Such channel debugging might be enabled via [slave_debug](#) on the channel, or via the `$U` flag in an [address * _ACCESS mapping table](#).)
- `TIMEOUT` (integer; default is 3600). Attempting to set a non-positive value will result in a value of 120 being used.
- `HOST` (hostname or IP address). A value must be specified for this option; (its presence is required).
- `MAX_PREPEND_INDEX` (integer; default is 1) (New in 8.0) Specifies the smallest index value that can be passed to [smfi_insheader](#) by the milter server and cause the resulting header field to be inserted at the top of the header block rather than the bottom.
- `PER_RECIPIENT_ACTIONS` (0 or 1; default is 0). (New in 7.0.5.33) Setting this option to 1 enables availability of Oracle's [milter extension SMFIF_SPECRCPT](#) for per-recipient modification actions.
- `PORT` (integer). A value must be specified for this option; (its presence is required).
- `PRESERVE_BREAKS` (0 or 1; default is 1) (New in Messaging Server 7.0.5) Preserve line breaks (line folding) in header lines during processing.
- `SESSION_INACTIVITY_TIMEOUT` (integer; default is 180). (New in 8.0) Time in session that a session is allowed to remain idle and still be a candidate for reuse.

- `SESSION_TIME` (integer; default is 3600). (New in 8.0) Maximum time, in seconds, that a single session can be used.
- `TRANSACTIONS_PER_SESSION` (integer; default is 100). (New in 8.0) Number of transactions allowed in a single session.
- `USE_JETTISON` (0 or 1; default is 0). (New in Messaging Server 7.0 update 2.) If this option is set to 1, then the Sieve "jettison" action will be used instead of "discard" if the milter calls for the message to be discarded. The default value of 0 causes "discard" to be used.
- `USE_QUIT_NC` (0 or 1; default is 0) (New in 8.0) Setting this option to 1 enables use of the [QUIT_NC milter command](#) so that sessions can be reused. This should only be set when the version of libmilter is recent enough to support the feature (Sendmail 8.14 or later).
- `QUARANTINE_ACTION` (string; default is "hold;") (New in 8.0.1) This option specifies the Sieve action to use when a milter quarantine message modifier is engaged. For example: `QUARANTINE_ACTION=require "fileinto"; fileinto "spam";` Milter quarantine actions always have an associated "reason" string. A `$R` can be used to substitute this string into the Sieve action. For example: `QUARANTINE_ACTION=require "reject"; reject "Message rejected, reason: $R";` A literal dollar sign in the Sieve action string must be doubled, e.g., `$$`. The default action that is performed if `QUARANTINE_ACTION` is not set is "hold;".

As of 8.0, support for milter connections via Socks has been removed, so the `SOCKS_HOST`, `SOCKS_PORT`, `SOCKS_USERNAME`, and `SOCKS_PASSWORD` milter options are no longer supported.

Some complex modifications of the milter spam filter plugin's behavior may be achieved using the [MILTER_MACROS mapping table](#).

Note that when using a Milter, the only relevant `spamfilterN*_action` options are `spamfilterN_null_action` (which has a proper default value) and `spamfilterN_string_action`; the `spamfilterN_action_M` and `spamfilterN_verdict_M` MTA options used with other sorts of spam/virus filter packages are *not* relevant with a Milter. And it is essential with Milter to also explicitly set the `spamfilterN_string_action` MTA option (to its special-for-Milters value of `data: , $M`) as the default value for `spamfilterN_string_action` is not appropriate for Milter use.

45.5 SpamAssassin spamfilterN_config_file

When using SpamAssassin, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter1_config_file /opt/SUNWmsgsr/config/spamassassin.dat
```

With SpamAssassin (supported as of JES MS 6.1), the file that this option names may contain the following options:

- `DEBUG` (0 or 1 up through JES MS 6.2; in JES MS 6.3, value is an integer and currently values of 0, 1, or 2 have meaning), default 0.
- `MESSAGE_BUFFER_SIZE` (integer), default 100000. New in JES MS 6.2. Specifies the maximum message size; larger messages will be truncated.

- HOST (hostname or IP address). A value must be specified for this option (its presence is required).
- PORT (integer), default 783.
- (New in 8.0) CONNECT_TIMEOUT (integer), defaults to the TIMEOUT option's value if TIMEOUT is greater than 0; if TIMEOUT is 0, then CONNECT_TIMEOUT defaults to 120. Attempting to set CONNECT_TIMEOUT to a non-positive value will result in a value of 60 being used. This option specifies the time, in seconds, that the MTA will wait to connect to SpamAssassin.
- (New in 6.2p5 and 6.3.) TIMEOUT (integer), default 3600 (seconds). This option controls how long the MTA will wait for a response from SpamAssassin. (Prior to JES MS 6.3 the timeout was hard-coded as 3600 seconds.) Note that the timeout for an initial connection to SpamAssassin is not controlled by this option, but rather is a setting for the TCP stack; this option instead controls the timeout for the MTA in getting back responses after the initial connection is established.
- SOCKS_HOST
- SOCKS_PORT (integer), default 1080.
- SOCKS_USERNAME
- SOCKS_PASSWORD
- MODE default 0. (Values 0, 1, 2, and (new in JES MS 6.2p1) 3 are supported.)
- FIELD default "Spam-Test".
- VERDICT
- USE_CHECK (0 or 1), default 1.
- USERNAME
- (New in JES MS 6.3p1.) USERNAME_MAPPING. This option is used to specify the name of a [mapping table](#) to probe with address information as the plugin receives recipient addresses from the MTA, to potentially override the USERNAME option's value. The probe format is:

current-username | current-recipient-address | current-optin-string

- Both the *current-optin-string* and the preceding vertical bar are omitted if no *optin* value was specified. If the mapping sets the \$Y flag, then the output string is taken to be the updated username (overriding the USERNAME SpamAssassin option value) to pass to *spamd*.

For instance, with SpamAssassin configured as spam/virus filter package number 2, and SpamAssassin option file settings that include

```
MODE=2  
FIELD=
```

and a setting that does an "addheader" action, such as

```
msconfig> set spamfilter2_string_action 'data:,addheader "Spam-test: $U";'
msconfig# show spamfilter2_string_action
role.mta.spamfilter2_string_action = data:,addheader "Spam-test: $U";
```

or in legacy configuration:

```
SPAMFILTER2_STRING_ACTION1=data:,addheader "Spam-test: $U"
```

could result in the addition of header lines such as:

```
Spam-test: False ; 0.1 / 4.5 ; FORGED_RCVD_HELO
Spam-test: True ; 12.9 / 4.5 ; RCVD_HELO_IP_MISMATCH,RCVD_IN_BL_SPAMCOP_NET,
RCVD_IN_XBL,RCVD_NUMERIC_HELO,URIBL_OB_SURBL,URIBL_SC_SURBL,URIBL_WS_SURBL
```

45.6 Archive spamfilterN_config_file

As of JES MS 6.3, the MTA supports archiving as a plug-in, similar to spam/virus filter package plug-ins. When using AXS:one, it.com, or Microsoft® Exchange Journal archiving, a [spamfilterN_config_file](#) MTA option might be set as

```
msconfig> set mta.spamfilter7_config_file /opt/SUNWmsgsr/config/archive.dat
```

With AXS:One, it.com or Microsoft Exchange Journal format use, the following options are available; see MTA configuration for archiving for further discussion.

- **DEBUG** (integer; default 0)
- **RESETDEBUG** (integer; default 0). (New in 8.0.) Enables or disables debug output on a per-transaction basis. If **RESETDEBUG** is nonzero, then [channel debugging](#) controls archiving debugging; If channel debugging is enabled, then the **RESETDEBUG** value becomes the debug level for the archiver; if channel debugging is disabled, then archiving debugging is disabled.
- **STYLE** (1-3; required). (Value 2 new in Messaging Server 7.0u1, value 3 new in 8.0.1.) The default value, 1, specifies AXS:One archiving. A value of 2 specifies it.com archiving and a value of 3 specified Microsoft Exchange Journal format archiving.
- **DIRECTORY** (string; no default). This option is required for AXS:One and it.com archives and is not used for Microsoft Exchange Journal format archiving. It specifies the directory where archive files are supposed to be written. In the case of it.com archives, `strftime` substitutions can be used to insert date and time information into the directory name.
- **DESTINATION** (string; no default; Microsoft Exchange Journal format only). If set, this option specifies the address where Exchange Journal format archive messages are to be sent. If the option is not set archive messages are sent to the various capture attributes associated with the message's authorized sender, envelope from, and envelope recipient addresses.
- **SOURCE_CHANNEL** (channel name; no default; required for Microsoft Exchange Journal format only). The **SOURCE_CHANNEL** option specifies the name of the [channel](#) used to submit Microsoft Exchange Journal format messages. It is recommended that a separate

channel be created for this purpose so that such submissions are clearly identifiable in the logs.

- **MODE** (integer; default 8%660). Sets the file creation mode used on all archive files that are created for AXS:One or it.com archives. Note that mode values can be written in the customary octal form by prefixing the value with "8%".
- **SUBDIRS** (integer; default 0). Specify how many subdirectories to use for AXS:One or it.com archives. Only values between 0 and 1000 will be used; attempting to set a value outside that range will result in 0 (no subdirectories) being used.
- **REVERSE** (0 or 1; default 1; AXS:One only).
- **USEHEADERRECIPIENTS** (0 or 1; default 0; AXS:One only). (New in MS 7.0.5).
- **TRUSTEXISTINGHASH** (0 or 1; default 0; AXS:One only).
- **IDSUFFIX** (string; default ""); AXS:One only). This option specifies a suffix to all MessageID fields generated in the AXS:One format. The default is the empty string. This option should be set to "-0100MD500" when running in operational mode and to "-0000MD500" when running in compliance mode.
- **POSTEDDATEMODE** (integer; default 100; AXS:One only). (New in Messaging Server 7.0u4). A value of 0 means to use the date/time from the Date: header field. The default value of 100 means to use an appropriate time, depending upon whether this is compliance *vs.* operational archiving: for compliance mode, use the start time of the archiving operation, whereas for operational mode, use the message's internal date. Any other value for **POSTEDDATEMODE** means to use a Received: header date/time.

45.7 Spamfilter early verdicts

Most spam/virus filter plugins base their decisions on message content. (SpamAssassin in particular acts solely based upon the message content it receives---though it attempts to make some assumptions about the message envelope based on material in the message itself.) However, as of Messaging Server 7.0 the MTA supports allowing spam/virus filters packages to return a so-called "early verdict", based upon the source IP address alone (as for instance in cases where the incoming connection is from a source IP that the spam/virus filter package considers to be a known spam source). Currently only the [Brightmail](#) and [milter](#) plugins are capable of returning such an early verdict. Early verdicts must be explicitly enabled in Brightmail; in milter, an early verdict corresponds to a message reject action taken at the SMFIC_CONNECT phase.

If the spam filter plugin is activated based on the source channel or the envelope from address, any early verdict checks are done at the start (MAIL FROM) of the SMTP transaction. However, if the spam filter plugin is activated based on destination channel or the recipient address, the check won't happen until that recipient address is communicated (RCPT TO). But in either case the rejection only occurs after the SMTP connection has been accepted by the Dispatcher and passed to the SMTP server.

In some cases it is preferable to have such checks done from the [Dispatcher](#) so that the connection itself can be refused. A mapping callout, `mm_check_reputation`, is therefore provided so this can be done from the [PORT_ACCESS mapping](#). The callout accepts two arguments separated by a vertical bar: (1) the slot number of the spam filter plugin to use, and (2) the IP address to check. The callout succeeds if an early verdict is returned.

An example of directly using Brightmail's "early verdict string" (without any additional MTA text, as would normally be added) is:

```
PORT_ACCESS
```

```
*|*|*|*          $:A$[IMTA_LIB:libimta.so,mm_check_reputation,1|$1]$N
```

The `:$:A` is used in this example to make sure this check is only done from the Dispatcher, and not the SMTP server. (In contrast, `:$:S` would be used to ensure that the check would be done only from the SMTP server and not from the Dispatcher.)

45.8 Milter implementation

The milter interface negotiates supports for various capabilities and actions. A given milter has the option of requiring a capability or action, optionally using a capability or action, or ignoring it entirely. This section documents what capabilities and actions are supported by Messaging Server as well as when support was first introduced.

The status of support for milter capabilities is:

Table 45.2 Milter capabilities

Capability	Description	Support Added
SMFIP_NOCONNECT	Skip SMFIC_CONNECT	6.3
SMFIP_NOHELO	Skip SMFIC_HELO	6.3
SMFIP_NOMAIL	Skip SMFIC_MAIL	6.3
SMFIP_NORCPT	Skip SMFIC_RCPT	6.3
SMFIP_NOBODY	Skip SMFIC_BODY	6.3
SMFIP_NOHDRS	Skip SMFIC_HEADER	6.3
SMFIP_NOEOH	Skip SMFIC_EOH	6.3
SMFIP_NR_HDR	Skip SMFIC_HEADER responses	7u4
SMFIP_NOUNKNOWN	Skip unknown commands	8.0
SMFIP_NODATA	Skip SMFIC_DATA	8.0
SMFIP_SKIP	MTA understands SMFIS_SKIP	7u4
SMFIP_RCPT_REJ	MTA should also send rejected RCPTs	8.0
SMFIP_NR_CONN	No reply for connect	8.0
SMFIP_NR_HELO	No reply for HELO	8.0
SMFIP_NR_MAIL	No reply for MAIL	8.0
SMFIP_NR_RCPT	No reply for RCPT	8.0
SMFIP_NR_DATA	No reply for DATA	8.0
SMFIP_NR_UNKN	No reply for UNKN	8.0

SMFIP_NR_EOH	No reply for end of header	8.0
SMFIP_NR_BODY	No reply for body chunk	8.0
SMFIP_HDR_LEADSPC	Header value leading space	8.0

The status of support for milter actions is:

Table 45.3 Milter actions

Action	Description	Support Added
SMFIF_ADDHDRS	Add headers	6.3
SMFIF_CHGBODY	Change body chunks	6.3
SMFIF_ADDRcpt	Add recipients	6.3
SMFIF_DELRcpt	Remove recipients	6.3
SMFIF_CHGHDRS	Change or delete headers	6.3
SMFIF_QUARANTINE	Quarantine message	6.3
SMFIF_CHGFROM	Filter may change from	8.0
SMFIF_ADDRcpt_PAR	Add recipients including args	unsupported
SMFIF_SETSYMLIST	Can send set of wanted macros	unsupported
SMFIF_SPECRCPT	Support per-recipient modification actions	7.0.5.33

The macros provided by the milter interface are:

Table 45.4 Available milter macros

Name	Protocol Phase	Content
<code>#{auth_authen}</code>	MAIL FROM	Authenticated sender address.
<code>#{auth_author}</code>	MAIL FROM	The value of the AUTH parameter to MAIL FROM. Added in 8.0.
<code>#{client_addr}</code>	CONNECT	The IP address of the SMTP client, expressed as a dotted quad value. Only set when SMTP over TCP is being used.
<code>#{destination_channel}</code>	RCPT TO	MTA destination channel for the current recipient.
<code>#{i}</code>	MAIL FROM	Queue id for the current message. The MTA generates a unique id for each session; this id is what appears in the <code>#{i}</code> macro.
<code>#{j}</code>	CONNECT	Text placed in the "by" clause of Received: header fields. 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.
<code>#{mail_addr}</code>	MAIL FROM	The MAIL FROM address for the current transaction.
<code>#{mail_host}</code>	MAIL FROM	The host part of the MAIL FROM address for the current transaction.
<code>#{optin}</code>	RCPT TO	Spamfilter optin value for the current RCPT TO address.
<code>#{rcpt_addr}</code>	RCPT TO	Current RCPT TO address.
<code>#{rcpt_host}</code>	RCPT TO	The host part of the current RCPT TO address.
<code>#{rcpt_mailer}</code>	RCPT TO	Set to "local" for valid recipient addresses, "error" for invalid addresses. New in 8.0.
<code>#{source_channel}</code>	MAIL FROM	MTA source channel.

The MTA's milter macros support can be extended/modified using the [MILTER_MACROS mapping table](#).

The `smfi_insheader` modification action associated with the `SMFIF_ADDHDRS` action flag specifies an index into the header where the field is to be inserted. Such semantics are not provided by Sieve; indeed, the `smfi_insheader` documentation itself notes that indices are not reliable. Prior to the 8.0 release, `smfi_insheader` was implemented by using a plain Sieve "addheader" for an index of 0 and "addheader :last" for a nonzero index. However, some milters, notably OpenDKIM, have been observed using an index value of 1 in an attempt to insert a field above the Received: field added by sendmail. Accordingly, a new milter spamfilter option, `MAX_PREPEND_INDEX`, has been added to deal with this and similar situations. `MAX_PREPEND_INDEX` specifies the smallest index value that can be passed to `smfi_insheader` by the milter server and cause the resulting header field to be inserted at the top of the header block rather than the bottom. The default value is 1.

The libmilter provided by sendmail 8.14 now supports milter session reuse for multiple SMTP sessions or transactions. Unfortunately this support does not appear to be negotiated, making it necessary to have an option to enable it in addition to various options to control its use. Accordingly, several new options have been added to the [milter spamfilter option file](#):

- `USE_QUIT_NC` (boolean, default 0) - Enable use of the `QUIT_NC` milter command so sessions can be reused. This should only be set when the version of libmilter is recent enough to support the feature. (Sendmail 8.14 or later.)
- `SESSION_INACTIVITY_TIMEOUT` (integer, default 180) - Time in session a session is allowed to remain idle and still be a candidate for reuse.
- `TRANSACTIONS_PER_SESSION` (integer, default 100) - Number of transactions allowed in a single session.
- `SESSION_TIME` (integer, default 3600) - Maximum time, in seconds, that a single session can be used.

As of the 8.0 release, proper remote port information is transferred through the `smfi_connect` callback.

Finally, support for milter connections via Socks has been removed, so the `SOCKS_HOST`, `SOCKS_PORT`, `SOCKS_USERNAME`, and `SOCKS_PASSWORD` [milter options](#) are no longer supported in 8.0 and later versions.

Most spam/virus filter packages return package-specific so-called "verdict strings", which the MTA is configured to interpret as desired (Sieve actions), with the correspondence controlled via pairs of MTA options `spamfilterN_verdict_M` and `spamfilterN_action_M`. However, milter normally returns an actual [Sieve scriptlet](#), which should be used verbatim. So the usual pairs of verdict/action MTA options are *not* used in the MTA's configuration for milter integration; instead, only the `spamfilterN_null_action` and `spamfilterN_string_action` MTA options are relevant for the MTA's milter configuration.

The default value for `spamfilterN_null_action`, namely `data: ,discard;`, is proper for use with milter, as if milter returns no verdict then the meaning is that the message should be discarded. To have milter's Sieve scriptlets used when milter *does* return a verdict, the `spamfilterN_string_action` option *must* be set to *exactly*:

```
data: , $M
```

So for instance:

```
msconfig> show spamfilter3_*
role.mta.spamfilter3_config_file = /opt/sun/comms/messaging64/config/miltertest.dat
role.mta.spamfilter3_library = /opt/sun/comms/messaging64/lib/libmilter.so
msconfig> show -default spamfilter3_null_action
role.mta.spamfilter3: data: ,discard
msconfig> set spamfilter3_string_action "data: , $M"
```

This setting of `spamfilter3_string_action` above is using the `$M` substitution (see the discussion of such substitutions in the discussion of the `spamfilter1_action_0` MTA option) which means to use the detailed verdict string provided by the milter directly as a Sieve scriptlet (and triggers special handling to allow proper handling of extra "long" returned verdicts, such as a milter-returned entire Sieve scriptlet).

45.8.1 MILTER_MACROS mapping table

Support for the `MILTER_MACROS` mapping table was added for Messaging Server 7.3-11.01. `MILTER_MACROS` is called by the milter [spam filter plugin](#) each time a [macro](#) is passed to the milter server. The probe format is:

```
spamfilter-index | command | macro-name | macro-value
```

Here `spamfilter-index` is an integer between 1 and 8 specifying the spam filter slot this milter is in, `command` is the command this macro precedes, so one of:

- `CONNECT`
- `MAIL`
- `RCPT`

The `macro-name` is simply the name of the macro being defined and `macro-value` is its value. Note that the `mapping_paranoid` MTA option, if set, will cause any vertical bar

characters that would have been in the macro-value field to be replaced by the specified character.

When the mapping returns, if \$N or \$F are set then the macro is dropped and never sent to the milter server. (This is also the behavior if \$Y or \$T is set, but with no additional string returned by the mapping template.)

If none of \$N, \$F, \$Y or \$T is set, then the original macro name and value are used, as if the MILTER_MACROS mapping table had not applied.

If, however, \$Y or \$T is set and a string is returned also, then the mapping's string result is processed as a series of macro name/value pairs, each name or value separated by vertical bars. Finally, if \$| is set in the mapping template in addition to \$Y or \$T, then only a single name-value pair is read from the result and the second and subsequent vertical bars are treated as part of the value.

Note that if either the original macro-value, or a replacement macro value returned as a string along with \$Y or \$T, includes a vertical bar character, |, then regardless of whether [mapping_paranoia](#) is used, the vertical bar test flag will be set (so that a mapping template may check for the original presence of a vertical bar in the macro name via a \$: | vs. \$; | test).

45.8.2 Milter single recipient extension

New in 7.0.5.33. Oracle has implemented a milter extension for per-recipient modification actions. This extension can be implemented in libmilter with the following changes:

- Add the following constant and routine declaration to include/libmilter/mfapi.h:

```
/* Oracle extension for recipient-specific modification actions */
#define SMFIF_SPECRcpt    0x1000000    // Recipient-specific modification actions

/*
** Specify a recipient for whom subsequent modification actions
** apply. Modification actions specified prior to this point will
** no longer apply to this recipient.
**
**      SMFICTX *ctx; Opaque context structure
**      char *rcpt; Null terminated list of null terminated envelope
**              recipient addresses subsequent actions apply to. These should
**              be in exactly the form passed to xxfi_envrcpt or the address
**              may not be selected for subsequent modification actions.
**
*/

LIBMILTER_API int smfi_speccrpt __P((SMFICTX *, char *));
```

- Add the following constant to include/libmilter/mfdef.h:

```
/* Oracle extension for recipient-specific modification actions */
#define SMFIR_SPECRcpt    'v'        /* Per-recipient modification actions */
```

- Add the following routine to libmilter/smfi.c:

```
/*
** SMFI_SPECRcpt -- Recipient-specific result (Oracle extension)
**
**      Parameters:
**          ctx -- Opaque context structure
```

imexpire invoking spamfilter packages

```
**          rcpt -- null terminated list of null terminated
**          recipient addresses
**
** Returns:
**          MI_SUCCESS/MI_FAILURE
**
*/

int
smfi_specrcpt(ctx, rcpt)
    SMFICTX *ctx;
    char *rcpt;
{
    size_t len, l;
    struct timeval timeout;
    char *ptr;

    if (rcpt == NULL || *rcpt == '\0')
        return MI_FAILURE;
    if (!mi_sendok(ctx, SMFIF_SPECRCPT))
        return MI_FAILURE;
    timeout.tv_sec = ctx->ctx_timeout;
    timeout.tv_usec = 0;
    len = 0;
    ptr = rcpt;
    do
    {
        len += (l = strlen(ptr) + 1);
        ptr += l;
    } while (*ptr != '\0');
    return mi_wr_cmd(ctx->ctx_sd, &timeout, SMFIF_SPECRCPT, rcpt, len);
}
```

In terms of the milter protocol, this extension consists of a single additional milter response:

```
**
*'v' SMFIR_SPECRCPT Specify recipient subsequent modification actions
    apply to.

char rcpt[][] List of NUL terminated recipients
```

The semantics are straightforward: SMFIR_SPECRCPT specifies a list of message recipients, and subsequent modification actions (SMFIR_ADDHEADER, SMFIR_INSHEADER, SMFIR_CHGHEADER, and SMFIR_QUARANTINE). The modification actions SMFIR_ADDRCPT and SMFIR_DELRCPT may be specified subsequent to an SMFIR_SPECRCPT, but are by nature not recipient-specific. SMFIR_REPLBODY is too expensive to implement on a per-recipient basis; its behavior subsequent to a SMFIR_SPECRCPT is undefined.

Because it's possible this extension may conflict with someone else's private extension, it must be explicitly enabled by setting the PER_RECIPIENT_ACTIONS option to 1 in the [milter plugin options file](#).

Since extensive code changes were required to implement this extension, it has not been incorporated into the normal libmilter.so library. A new libmilters.so library containing the extension has been provided instead.

45.9 imexpire invoking spamfilter packages

New in Messaging Server 7.0.5, the `imexpire` utility has a new `channel` attribute to specify the name of an [MTA channel](#). When this attribute is used, any [source channel spam/virus filter package configured on that channel](#) will be applied to each message that is scanned by `imexpire`, and any [spamadjust](#), [spamttest](#), [virusset](#), [virustest](#), or [added headers](#) will then be visible to the [Sieve expression](#) used to expire messages. (Of course, use of Sieve expressions to expire messages must also be enabled via the [expiresieve](#) Message Store option.)

The MTA options [scan_channel](#), [scan_originator](#), and [scan_recipient](#) may be used to establish context (Sieve values) for non-channel evaluations of Sieve filters, such as `imexpire` invocations of spam/virus filter packages, though note that `scan_channel` is not needed for `imexpire`'s spamfilter package invocation case (since in such a case `imexpire`'s `channel` attribute is used to set the MTA channel).

For example, suppose that MTA channel invokes SpamAssassin, which is then configured to perform a [spamadjust](#) to communicate its results. In this scenario, an expression of the form

```
require ["comparator-i;ascii-numeric", "relational", "spamttest"];
spamttest :value "ge" :comparator "i;ascii-numeric" "5";
```

should expire any message that received a spam score of 5 or more.

The new-in-7.0.5 `rescanhours` attribute is also especially relevant when using `imexpire` to perform post-delivery spam/virus filtering. `rescanhours` tells `imexpire` to rescan those messages that have not been scanned for the specified number of hours.



Chapter 46 MeterMaid

46.1 MeterMaid options	46-1
46.1.1 enable Option Under metermaid	46-2
46.1.2 debug Option Under metermaid_client	46-2
46.1.3 local_table	46-2
46.1.4 remote_table	46-3
46.1.5 metermaid_client	46-4
46.1.6 async Option	46-4
46.1.7 backlog Option Under metermaid	46-4
46.1.8 connecttimeout Option Under metermaid_client	46-4
46.1.9 max_conns Option Under metermaid_client	46-4
46.1.10 max_conns Option Under remote_server	46-4
46.1.11 maxthreads Option Under metermaid	46-4
46.1.12 server_host Option Under metermaid_client	46-4
46.1.13 server_host Option Under remote_server	46-5
46.1.14 server_port Option Under metermaid_client	46-5
46.1.15 server_port Option Under remote_server	46-5
46.1.16 server_nickname Option	46-5
46.1.17 service Option	46-5
46.1.18 port Option Under metermaid	46-5
46.1.19 secret Option Under metermaid	46-5
46.1.20 timeout Option Under metermaid_client	46-5
46.1.21 listenaddr Option Under metermaid	46-5

MeterMaid is a facility that comprises a server, that maintains "tables" of data, and a client side (in particular, the MTA's callouts to MeterMaid via mapping table MeterMaid routine callouts, [metermaid: URLs](#) encoded into MTA configuration, and (new in Messaging Server 7.2) [Sieve "metermaid" tests or actions](#)). The MeterMaid server maintains its data in-memory; this offers high performance, but note that it does imply that the MeterMaid data is *not* preserved across MeterMaid (or Messaging Server as a whole) restarts. As multiple processes can communicate with the MeterMaid server over protocol, MeterMaid permits across-process tracking of data. MeterMaid is thus particularly suited for configuring "throttle" effects.

MeterMaid requires configuration of itself -- the MeterMaid server, and some basics of MeterMaid client operation -- configured in Unified Configuration via [MeterMaid options](#), or in legacy configuration via configutil parameters. Once MeterMaid's own operation is established, then configuring the MTA on how to find/communicate with MeterMaid is configured via [MeterMaid MTA options](#). And then any specific MeterMaid uses may be configured into the MTA via [mapping table routine callouts](#), [metermaid: URLs in appropriate MTA configuration options](#), or use of [Sieve "metermaid" tests or actions](#).

For a number of examples of MeterMaid use in the form of MTA mapping table [callouts to MeterMaid routines](#), see the discussion of [Triggering effects from transaction logging with LOG_ACTION](#).

46.1 MeterMaid options

To find all options potentially relevant to [MeterMaid](#), try doing

```
msconfig> apropos metermaid
```

Note that there are the options relevant for the MeterMaid server, settable under the `metermaid` group, and there are options relevant for any MeterMaid clients, settable under `metermaid_client` group; respectively, these correspond to the legacy configuration `metermaid.*` and `metermaid.mtaclient.*` configutil parameters. Some options are settable either generally for a named `metermaid` or `metermaid_client` group, or settable specifically for a named table under a `local_table` or `remote_table` group. Then there are also a number of **MTA options** that override, for MTA purposes, some of the normal `metermaid` or `metermaid_client` options. There is also the `metermaidtable` IMAP option (under the `pwexpirealert` group), to specify what MeterMaid table to use for password expiration alerts.

A named `remote_table` group may only be set under `metermaid_client`. A named `local_table` group may be set either under `metermaid` or `metermaid_client`.

46.1.1 enable Option Under metermaid

The `enable` MeterMaid option enables the **MeterMaid** service on `start-msg` startup.

46.1.2 debug Option Under metermaid_client

The `debug` MeterMaid Client option enables debug output from the MTA client into SMTP log files.

46.1.3 local_table

Under a MeterMaid named `local_table` group, there are a number of options that may be set. *E.g.*,

```
msconfig> show -default metermaid.local_table:table-name.quota
role.metermaid.local_table:table-name.quota: 100
```

46.1.3.1 data_type Option

The `data_type` MeterMaid `local_table` option specifies the type of data to be stored in this table: one of `ipv4` or `string`.

46.1.3.2 block_time Option

The `block_time` MeterMaid `local_table` option specifies an initial period for greylisting during which requests will be blocked. It takes an argument in either the standard **ISO 8601 P format**, specifying the duration of the period, or a plain integer argument which will be interpreted as a number of seconds for the period.

46.1.3.3 resubmit_time Option

The `resubmit_time` MeterMaid option specifies the period for greylisting during which a request must be received again in order to be permitted in the future. It takes an argument in either the standard **ISO 8601 P format**, specifying the duration of the period, or a plain integer argument which will be interpreted as a number of seconds for the period.

46.1.3.4 inactivity_time Option

The `inactivity_time` MeterMaid option specifies a period for greylisting during which a resubmitted entry will remain 'known' to MeterMaid. It takes an argument in either the standard [ISO 8601 P format](#), specifying the duration of the period, or a plain integer argument which will be interpreted as a number of seconds for the period.

46.1.3.5 max_entries Option

The `max_entries` MeterMaid `local_table` option specifies the maximum number of entries to maintain in this table.

46.1.3.6 table_options Option

The `table_options` MeterMaid `local_table` option specifies a space-separated list of options for this table; can include `penalize` or `nocase` (for string data).

46.1.3.7 quota Option

The `quota` MeterMaid `local_table` option specifies the number of connections to permit per `quota_time` time period.

46.1.3.8 quota_time Option

The `quota_time` MeterMaid `local_table` option specifies the period of time to allow `quota` number of connections. It takes an argument in either the standard [ISO 8601 P format](#), specifying the duration of the period, or a plain integer argument which will be interpreted as a number of seconds for the period.

46.1.3.9 storage Option

The `storage` MeterMaid `local_table` option specifies the method of data storage for this table: one of `hash` or `splay`.

46.1.3.10 table_type Option

The `table_type` MeterMaid `local_table` option specifies the type of table; valid selections include `throttle`, `simple`, or `greylisting`.

46.1.3.11 value_type Option

When using a `'simple'` table, the `value_type` MeterMaid `local_table` option specifies the kind of data used for the values in this table. Currently one may choose between `integer` or `string`.

46.1.4 remote_table

A `remote_table` named group may be set under `metermaid_client`, to name a table maintained by that name on a remote MeterMaid server. New in MS 8.0, the `server_nickname` option may be set under a named `remote_table` group. No further options are currently settable under `remote_table`; table options would instead be set under the `local_table` definition on the remote MeterMaid server.

46.1.5 metermaid_client

A MeterMaid client can be run on any Messaging Server system to talk to (connect to) a MeterMaid server (which may be running on a different system) maintaining various tables of information. That is, a MeterMaid client is a consumer/updater of information maintained by a MeterMaid server.

In Unified Configuration, any desired options for MeterMaid client operation are set under

```
msconfig> set metermaid_client.connectfrequency 10
```

See the [MeterMaid](#) topic for further discussion of general MeterMaid operation. And see [metermaid](#) for discussion of MeterMaid server options set under metermaid.

46.1.6 async Option

The `async` MeterMaid option sets whether [MeterMaid](#) should use asynchronous thread scheduling (default) or the new, experimental linear thread scheduling.

46.1.7 backlog Option Under metermaid

The `backlog` [MeterMaid option](#) specifies the number of connections to permit to be established in the TCP listen queue.

46.1.8 connecttimeout Option Under metermaid_client

The `connecttimeout` MeterMaid Client option, (`metermaid_client.connecttimeout` in Unified Configuration or `metermaid.mtaclient.connectwait` in legacy configuration), specifies how long a thread should wait for a connection to be established to [MeterMaid](#) (seconds).

46.1.9 max_conns Option Under metermaid_client

The `max_conns` MeterMaid client option, `metermaid_client.max_conns`, specifies how many concurrent connections can be established to MeterMaid from a single process.

46.1.10 max_conns Option Under remote_server

The `max_conns` MeterMaid client `remote_server` option, `metermaid_client.remote_server:server-name.max_conns`, specifies how many concurrent connections can be established to the specified remote server from a single process.

46.1.11 maxthreads Option Under metermaid

The `maxthreads` [MeterMaid option](#) specifies the maximum number of work threads.

46.1.12 server_host Option Under metermaid_client

The `server_host` MeterMaid Client option, (`metermaid_client.server_host` in Unified Configuration or `metermaid.config.serverhost` in legacy configuration), specifies the host or IP address of the MeterMaid server to use.

46.1.13 server_host Option Under remote_server

The `server_host` option for a `remote_server` specifies the host or IP address of the remote MeterMaid server to use.

46.1.14 server_port Option Under metermaid_client

The `server_port` MeterMaid client option specifies the TCP port to connect to when contacting the [MeterMaid](#) server. If this option is not specified but a local MeterMaid server is enabled, then the TCP port listened on by the local MeterMaid will be assumed. If no local MeterMaid server is running, then the default value of 63837 is used.

46.1.15 server_port Option Under remote_server

The `server_port` MeterMaid `remote_server` option specifies the TCP port to which the `metermaid_client` should connect for this `remote_server`.

46.1.16 server_nickname Option

The `server_nickname``metermaid_client.remote_table` option specifies the name of the `remote_server` entries that define a remote server.

46.1.17 service Option

The legacy configuration `service` channel option has been replaced in Unified Configuration by [serviceconversion](#).

46.1.18 port Option Under metermaid

The `port` MeterMaid option specifies the TCP port number on which [MeterMaid](#) listens for connections.

46.1.19 secret Option Under metermaid

The `secret` MeterMaid option specifies the secret used to authenticate [MeterMaid](#) clients with the server.

46.1.20 timeout Option Under metermaid_client

The `timeout` MeterMaid Client option, (`metermaid_client.timeout` in Unified Configuration, or `metermaid.mtaclient.readwait` in legacy configuration), specifies how long in seconds a MeterMaid client will wait for communication with [MeterMaid](#).

46.1.21 listenaddr Option Under metermaid

The `listenaddr` metermaid option specifies the IPv4 address on which [MeterMaid](#) should listen. If unset specifically for MeterMaid, MeterMaid defaults to the base value of `listenaddr` (`service.listenaddr` in legacy configuration).

The allowed values for this option include an IPv4 address in dotted decimal form (e.g., 127.0.0.1), or a short form or fully-qualified DNS host name which will be resolved to an IPv4

address by obtaining the DNS A record for the name. To explicitly specify the default value of binding to all available interfaces, the string "INADDR_ANY" may be used. To bind to the loopback device, 127.0.0.1, the string "localhost" may be specified.

Chapter 47 Notification messages

47.1 Notification message types	47-1
47.1.1 Message Store notifications that a user himself is overquota	47-3
47.2 Notification message generation timing	47-3
47.3 Notification message format	47-4
47.3.1 DSN language and customization	47-8
47.3.2 MDN language and customization	47-17
47.3.3 NOTIFICATION_LANGUAGE and DISPOSITION_LANGUAGE sample mapping tables	47-21
47.4 Notification message routing	47-22
47.5 Bounces of spam messages	47-23
47.6 Notification message logging	47-24
47.7 Message size limits and notification messages	47-25
47.8 Postmaster addresses	47-25

The defining characteristic of notification messages is that they have an empty envelope From address. [Notification message types](#) gives an overview of the different types of notification messages, those generated by the MTA itself, as well as those notification messages generated externally and sent to the MTA.

Since notification messages (mostly) are generated automatically, there are some special aspects to them. [Notification message generation timing](#) discusses the timing of generation, and for some types of notification messages the delivery scheduling, for notification messages generated by the MTA. The special case of notification messages generated by the Message Store is briefly discussed in [Message Store notifications that a user himself is overquota](#); see Message Store documentation for further details. The format, and potential language selection and customization options, of those notification messages generated by the MTA are discussed in [Notification message format](#). Special handling of notification messages may sometimes be desirable; [Notification message routing](#) discusses the routing of notification messages, and [Bounces of spam messages](#) in particular discusses the case of of spam "blow back" notification messages. [Notification message logging](#) discusses special features and factors in MTA message transaction log entries relating to notification messages, and [Message size limits and notification messages](#) discusses message size limits in relation to notification messages.

47.1 Notification message types

The MTA may generate notification messages itself automatically. These may be Delivery Status Notifications (see RFCs 3461-3464, which are updates to RFCs 1891-1894), such as a notification of a successful message delivery (a so-called "delivery receipt"), or a warning to the original message sender that the delivery of their message has been delayed or has failed. The MTA can also optionally (see the `*warnpost` and `*sendpost` channel options) generate notifications to the postmaster regarding failed and/or delayed message delivery; such a notification to the postmaster is more-or-less a copy of the notification going back to the original message sender. (In particular, note that the postmaster gets a copy in the same language chosen based upon the original sender's language selection.)

Or MTA-generated notification messages may be Message Disposition Notifications (see [RFC 3798](#), which updated [RFC 2298](#)) generated due to Sieve filter actions such as `vacation`. Note that a common case of Message Disposition Notifications is the case of so-called "read receipts", which in Message Disposition Notifications correspond to a disposition of

"displayed" (since whether or not a user actually *read* a message is a subjective question for the user, whereas the *display* of a message is something mail user agent software can detect); in any case, such "read receipt" notifications are generated by end user mail clients, *not* by the MTA (which merely relays them as with any other message).

The MTA will also automatically generate notification messages (DSN format messages of "error" type) to report certain sorts of syntax errors. Syntax errors in [Sieve filters](#) will be reported to the "responsible" address via a notification message: syntax errors in [channel level Sieve filters](#) or the system Sieve filter [systemfilter](#) (or in legacy configuration, the `SERVERROOT/config/imta.filter` file, located prior to MS 7.0.5 via the `imta_system_filter_file` MTA Tailor option) will be reported to the postmaster; syntax errors in user Sieve filters will be reported to the individual user to whom the Sieve filter belongs; syntax errors in "head of household" (also called "parental control") [Sieve filters](#) will be reported to the head of household filter "owner" (see the [ldap_hoh_owner MTA option](#)).

Similarly, (as of JES MS 6.1) the MTA will report syntax errors in alias or group definitions (that is, errors of non-existent putatively "local" users in the alias/group membership, or clearly syntactically invalid addresses in the alias/group membership) to the entire alias/group membership. These error reports are in DSN format. Note: (a) There is a critical, fundamental distinction between a mailing list (where errors in list definition as well as problems with list message deliveries get reported *only* to the list "owner" as specified via an [mgrpErrorsTo](#) attribute overriding the original message envelope From address) *vs.* a group (which has no [mgrpErrorsTo](#) attribute and hence retains the original message envelope From address). From the MTA point of view, a group is merely a--possibly large--alias--an "auto-forwarder" in Internet e-mail terms. (b) The group definition syntax errors reported to the entire group membership are not message delivery failures--which in the case of a group would be reported merely to the original message sender--but rather are those syntactic errors in the group definition which are apparent to the MTA at group alias expansion time. Because a group, when properly used, is a set of aliases for a single person or small, closely related set of people, problems with the group membership definition are considered problems with the alias setup that the rest of the group members should be informed about so that they can fix the definition.

The MTA will also generate messages that have the form of notification messages (they have an empty envelope From and contain the original message as an encapsulated part, usually perceived as an "attachment") in response to typical forms of [capture](#) configuration: that is, when a "capture" attribute (see the [ldap_capture MTA option](#)) is set on a user, or when an explicit [Sieve "capture" action](#) applies to a message, or when "capture" is triggered via another mechanism such as an address-based [*_ACCESS mapping table "capture" flag \(\\$M\)](#). The capture message, having the superficial form of a notification message, will be sent to the address specified to receive the capture copies. Though the form of such capture copies is similar to other sorts of notification messages, the intended purpose is usually quite different, as [capture of messages](#) is usually intended either for monitoring (of a user's e-mail) or archiving purposes.

The MTA may also issue SMTP level rejections of attempted message submissions. In such cases, the MTA is not generating the notification *message* itself; in such cases generation of a notification message is the responsibility of the SMTP client attempting to send the message. (And the SMTP client may or may not include the MTA's actual SMTP rejection text in the message text that it generates to display or send to the original message sender.)

The Message Store can generate "notifications" that a user is over quota; such quota "notifications" are deposited directly into the Message Store (without passing through the MTA at all); see [Message Store notifications that a user himself is overquota](#).

Also, other non-MTA components of Messaging Server (such as `msprobe`) may have capabilities for generating alarm messages "from" some form of postmaster address and "to" some form of postmaster address; see the [noticesender](#) and [noticercpt](#) [Alarm options](#).

Furthermore, the MTA also processes numerous notification messages generated by systems other than the MTA -- notification messages that arrive in to the MTA like any other messages.

All of these cases involve different issues, and different configuration choices. Whenever a question about a notification message arises, it is critical to first determine what type of notification message is involved. Note that looking at the **outermost header of the notification message** is the **most important first step** in determining what type of notification message one is dealing with.

47.1.1 Message Store notifications that a user himself is overquota

Notifications to a user that that user him or herself is overquota are generated by the Message Store (if the [quotanotification](#) Message Store option is enabled), or generated due to the system administrator using the `imquotacheck` utility to manually generate such notifications; they are not generated by the MTA, and, since such warnings are deposited directly into the store bypassing the MTA, they do not even go through the MTA. (Thus note that the warnings to a local user of that selfsame user being overquota, or near to overquota, are a separate and *very* different case than an MTA-generated notification telling a message sender that his/her message could not be delivered to its intended recipient with the reason for nondelivery happening to be that that intended local recipient was overquota.)

Such warnings to a user that he/she himself/herself has gone overquota, or is near going overquota, are configurable via a number of Message Store `quota*` options ([store.quota*](#) options in Unified Configuration, corresponding to legacy configuration `store.*quota*` parameters and `local.store.*quota*` configutil parameters), as well as `imquotacheck` features. The handling of quota is a complex subject in itself; please see the *Administration Guide* for much more complete discussion, as the discussion here is simplified and merely attempts to give an (over-simplified) orientation.

The [quotanotification](#) Message Store option controls whether the Message Store generates quota-related notifications to the user. The reporting of user overquota status normally is triggered by a user's mailbox reaching the threshold specified by the [quotawarn](#) option (legacy configuration `store.quotawarn` configutil parameter). However, if [quotaoverdraft](#) is set, (legacy configuration `local.store.quotaoverdraft=on`), then notifications are not generated until the user's mailbox actually exceeds their specified quota.

Besides depositing a warning message into a user's mailbox when that user first goes over the quota notification threshold, for users of IMAP clients that support the IMAP ALERT functionality, the warning message will be displayed on the user's client each time the user selects a mailbox. Also, the [quotaexceededmsginterval](#) Message Store option (in legacy configuration, the configutil parameter `store.quotaexceededmsginterval`) controls the periodic sending of additional overquota warnings if a user remains overquota.

47.2 Notification message generation timing

Many [sorts of notification message](#) the MTA generates immediately whenever the relevant type of event occurs. For instance, if a message suffers a permanent rejection upon a delivery attempt, then the MTA will immediately generate a non-delivery report (often referred to as

a "bounce message") to send back to the original message sender. Notifications generated due to a Sieve ["vacation"](#) or ["capture"](#) action are generated when such a Sieve filter is evaluated (when an original message that triggers the Sieve ["vacation"](#) or ["capture"](#) action is processed). Similarly, notifications warning of a [Sieve syntax error](#) are generated whenever the relevant Sieve filter is processed in an attempt to apply it to an incoming original message.

Another trigger for MTA generation of notification messages is postmaster use of the [imsimta return utility](#) or [imsimta qm](#) utility's `return` command to explicitly, immediately bounce specified messages.

But notification messages regarding temporary delivery problems, or those reporting a message delivery failure due to finally "timing out" after repeated delivery attempts, are instead generated upon a periodic (configurable) schedule; see the [*notices](#) channel options for configuration of eligibility for such notifications, and the [MTA return_job](#)'s [scheduling](#) for actual generation of such notifications. Similarly, [Message-Store-generated user overquota warnings](#) may be issued periodically; see the [quotaexceededmsginterval](#) Message Store option (legacy configuration `store.quotaexceededmsginterval` `configutil` parameter).

The reporting of delayed (or eventually timed-out-and-given-up-on) messages is triggered by the [MTA return_job](#), which checks the settings of any [*notices](#) channel options in order to decide whether it is time to generate a notification message regarding the delayed (or eventually failed due to timing out) message. The [MTA return_job](#) must thus be [scheduled](#) as appropriate for a site's needs in relation to the generation of such notification messages. As of JES MS 6.0, the Messaging Server [Scheduler](#), `imsched`, is normally used to schedule the running of the MTA's `return_job`. (In previous versions, such scheduling was normally performed by the [Job Controller](#) via a `PERIODIC_JOB` definition---an approach that is now deprecated.) In Unified Configuration, the Scheduler configuration, besides its [enable](#) option, consists primarily of a [crontab](#) option setting for each scheduled job. In particular, in Unified Configuration the [MTA return_job](#) schedule is set via the `schedule.task:return_job.crontab` option setting:

```
msconfig> show task:return_job.crontab
role.schedule.task:return_job.crontab = 30 0 * * * lib/return_job
```

In legacy configuration, the Scheduler configuration is controlled by `configutil` parameters; for the MTA's `return_job`, see in particular the `local.schedule.return_job` parameter, whose default value is

```
30 0 * * * /opt/SUMWmsgsr/lib/return_job
```

This is UNIX crontab format, *i.e.*,

minutes-after-hour hour day-of-month month-of-year day-of-week script

So the normal setting corresponds to the `return_job` being set to run at 30 minutes after midnight every day, (with the asterisks indicating, respectively, every day of the month, every month of the year, every day of the week).

47.3 Notification message format

The MTA generates standard notification messages (DSNs and MDNs) in the formats defined by the respective Internet standards, in particular [RFC 3462](#) and [RFC 3464](#) in the case of DSNs,

and [RFC 3798](#) in the case of MDNs. Shown in [Example non-delivery DSN](#) is a sample non-delivery DSN with annotations.

Example non-delivery DSN

```

Return-Path: <> (1)
Received: from process-daemon.host1.domain.com by host1.domain.com (Sun Java (2)
  System Messaging Server 6.2-3.04 (built Jul 15 2005)) id
  <0JRJ00301JFFR300@host1.domain.com> for user1@domain.com (ORCPT
  user1@domain.com); Thu, 15 Nov 2007 01:25:45 -0800 (PST)
Received: from host1.domain.com (Sun Java System Messaging Server 6.2-3.04
  (built Jul 15 2005)) id <0JRJ0038FJIWYS00@host1.domain.com> for (3)
  user1@domain.com (ORCPT user1@domain.com); Thu, 15 Nov 2007 01:25:45 -0800
  (PST)
Date: November 15, 2007 1:25:45 AM -0800 (PST)
From: Internet Mail Delivery <postmaster@host1.domain.com> (4)
Subject: Delivery Notification: Delivery has failed (5)
To: user1@domain.com
Message-Id: <0JRJ0038MJIXYS00@domain.com>
Mime-Version: 1.0
Content-Type: multipart/report; (6)
  boundary="Boundary_(ID_7qPwt/LotX5gyloVEHL7SQ)"; report-type=delivery-status
Original-Recipient: rfc822;user1@domain.com

--Boundary_(ID_7qPwt/LotX5gyloVEHL7SQ)
Content-type: text/plain; charset=us-ascii (7)
Content-language: en-US
Content-transfer-encoding: 7BIT

This report relates to a message you sent with the following header fields: (8)

  Message-id: <01MZZ7DIEUG400LXL0@host1.domain.com> (9)
  Date: Thu, 15 Nov 2007 01:23:18 -0800 (PST)
  From: user1@domain.com
  To: bogus@remote.com
  Subject: test to generate a bounce -- please ignore

Your message cannot be delivered to the following recipients: (10)

  Recipient address: bogus@remote.com (11)
  Original address: bogus@remote.com (12)
  Reason: Remote SMTP server has rejected address
  Diagnostic code: smtp;550 5.1.1 <bogus@remote.com>... User unknown
  Remote system: dns;mx1.remote.com (TCP|129.146.11.74|42429|129.156.85.165|25)
  (sunmail5.uk.sun.com ESMTP Sendmail 8.13.8+Sun/8.13.7/ENSMAIL,v2.2;
  Thu, 15 Nov 2007 09:25:44 GMT)

--Boundary_(ID_7qPwt/LotX5gyloVEHL7SQ)
Content-type: message/delivery-status (13)

Original-envelope-id: 01MZZ7DIEUG400LXL0@host1.domain.com
Reporting-MTA: dns;host1.domain.com (tcp-daemon) (14)
(15)
Original-recipient: rfc822;bogus@remote.com (16)
Final-recipient: rfc822;bogus@remote.com (17)
Action: failed
Status: 5.1.1 (Remote SMTP server has rejected address)
Remote-MTA: dns;mx1.remote.com (TCP|129.146.11.74|42429|129.156.85.165|25)
  (sunmail5.uk.sun.com ESMTP Sendmail 8.13.8+Sun/8.13.7/ENSMAIL,v2.2; Thu, 15

```

```
Nov 2007 09:25:44 GMT)
Diagnostic-code: smtp;550 5.1.1 <bogus@remote.com>... User unknown

--Boundary_(ID_7qPwt/LotX5gy1oVEHL7SQ)
Content-type: message/rfc822

Return-path: <user1@domain.com> (18)
Received: from [129.158.87.66] by host1.domain.com (Sun Java System Messaging (19)
Server 6.2-3.04 (built Jul 15 2005)) with ESMTPA id
<01MZZ7D1WINK00LXL0@host1.domain.com> for bogus@remote.com (ORCPT
bogus@remote.com); Thu, 15 Nov 2007 1:25:08 -0800 (PST)
Date: Thu, 15 Nov 2007 01:23:18 -0800 (PST)
From: user1@domain.com
Subject: test to generate a bounce -- please ignore
To: bogus@remote.com
Message-id: <01MZZ7DIEUG400LXL0@host1.domain.com>
MIME-version: 1.0
Content-type: TEXT/PLAIN
Content-transfer-encoding: 7BIT

test

--Boundary_(ID_7qPwt/LotX5gy1oVEHL7SQ)--
```

1. As with all standard notification messages, the envelope From is empty, as shown/recorded here in the Return-path: header line.
2. By default, notification messages are enqueued to the [process channel](#); but see the [notificationchannel](#) channel option. Note that the Received: header line below this one---the very "first" (in time) Received: header line---corresponds to the initial enqueue of the notification message from the channel that determined that a notification message was needed to the process channel. (If the initial channel that determined that a notification message was needed had a [notificationchannel](#) specified other than the process channel, the initial enqueue would instead have been to that alternate [notificationchannel](#).) Thus by default (no [notificationchannel](#) used) this second-from-the-bottom Received: header line, this second-from-oldest Received: header line, on any notification message generated by the Messaging Server MTA, will show the notification message coming from the process channel.
3. Note the "for *recipient*" clause present in this Received: header line. That such a clause is present shows that at this point in time, the message had only one recipient, and that the (default) [receivedfor](#) channel option was in effect. Note that the fact that there was only one recipient for the message at that point (initial enqueue to the [process channel](#)), indicates that the postmaster was not being copied on this notification: that one of [nosendpost](#) or [errsendpost](#) was in effect.
4. By default, the MTA-wide postmaster address is used in the From: header line for notifications; see the [return_address](#) and [return_personal](#) MTA options. But override postmaster addresses may be set on a per-channel basis (see the [returnaddress](#) and [returnpersonal](#) channel options), or on a per-domain basis (see the [mailDomainReportAddress](#) attribute, or more precisely, the attribute named by the [ldap_domain_attr_report_address](#) MTA option), or on a per-message basis via the [FROM_ACCESS](#) mapping table's \$ (or \$) flags.
5. The Subject: field for DSNs has the default text shown in [Table of DSN types and their default Subject: field text](#). The default may be overridden for all types of DSNs using the

[SUBJECT](#) option in the [return_option.opt](#) file, or may be overridden on a per-type-of-DSN basis using the `$T` flag of the [NOTIFICATION_LANGUAGE](#) mapping table.

6. DSNs at the outermost MIME level have type:


```
Content-type: multipart/report; ... report-type=delivery-status
```
7. The MIME header lines for the first message part (the human-readable part) are controlled by the [return_prefix.txt](#) file . Which language-specific [return_prefix.txt](#) file is used may be controlled by the [NOTIFICATION_LANGUAGE](#) mapping table .
8. This line (or optionally multiple lines) of introductory text is also set in the [return_prefix.txt](#) file. Normally, [return_prefix.txt](#) also includes a `%H` substitution to cause insertion of (a sample of) the original message headers.
9. The [return_header.opt](#) file controls exactly which header lines a `%H` substitution inserts. The `%H` substitution itself is located in the language-specific [return_prefix.txt](#) file.
10. This line of text is set in the appropriate [return_*.txt](#) file , corresponding to the type of DSN being generated: `failed`, `bounced`, `timedout`, `delayed`, `deferred`, `delivered`, `read`, `relayed`, `expanded`, `capture`, or `error`. In the case of this sample DSN, this is a "failed" DSN (the original message could not be delivered), so the [return_failed.txt](#) file sets the line of text, as well as causing insertion of a list of the message's recipients via a `%R` substitution.
11. If the relevant [return_*.txt](#) file requested it via a `%R` substitution, then a description of the recipients of the original message (and what happened for each such recipient) will be included. (Note tht the exact form of recipient address reported as "Recipient address:" may be affected by the setting on the source channel -- the channel generating the notification -- of [includefinal](#), [suppressfinal](#), or [useintermediate](#).) Overall, this is intended as a more human-readable version of the information also presented (in standard, machine-readable form) in the second part of the DSN, at (15). In particular, the text labels for each of these fields (*e.g.*, " Recipient address: "---note the two leading spaces as well as the terminal colon and trailing space are considered part of the label) are configurable via the language-specific [return_option.opt](#) file (selected via the [NOTIFICATION_LANGUAGE](#) mapping table).
12. Optionally, additional, alternate text can be configured per SMTP enhanced status code (*e.g.*, in this example 5.1.1); if such alternate text has been configured in the [return_option.opt](#) file corresponding to the enhanced status code being reported, then it would be presented after the "Original address: " (ORIGINAL_ADDRESS) line and before the "Reason: " (REASON) line.
13. This second part of a DSN is a machine-readable part. Note that unlike the first, human-readable part of a DSN, see (7), the machine-readable part is *not* subject to customization/localization/alternate language selection. This part uses the format and fields specified in [RFC 3464 \(An Extensible Message Format for Delivery Status Notifications\)](#). Because this part is in a precisely specified, machine-readable format, clients or gateways that choose to do so can in principle make choices of their own on how and whether to present this part to users, potentially including performing their own (client or gateway level) translation of the (precisely defined) content of this part.
14. The reporting MTA (the MTA that is generating the notification message) is reported here. Note that this is not necessarily the MTA that rejected the message: as in the case of this

example, where the rejection occurred at the SMTP protocol level, with the message being rejected by a remote host, but where the "local" MTA then had responsibility for generating the notification message. Note also that the Received: header lines on the DSN itself--in particular the first (in time, so lowest in the header) Received: header line---also show you what MTA generated the DSN. But whichever place you gather this piece of information from, note that it is a *critical* piece of information: you can only [customize](#) the DSNs that *your* system generates!

15. Some additional fields can potentially appear here, including X400-Content-identifier:, Content-identifier:, and UA-content-id:. As of JES MS 7.0, Arrival-date: and Future-release-request: also may potentially appear.
16. A set of information fields is output for each of the recipients of the original message, describing what happened for that recipient.
17. The exact form of recipient address reported as the "Final-recipient:" may be affected by the setting on the source channel (the channel generating the notification) of [includefinal](#), [suppressfinal](#), or [useintermediate](#).
18. The Return-path: records the envelope From of the original message.
19. The third and final part of the DSN contains the original message (or merely the headers of the original message, if the sender originally set the NOTARY non-return-of-content flag, or if the MTA was obliged to set or perform non-return-of-content due to message size restrictions). The entire header of the original message is included.

The MTA generates "autoreply" or "vacation" messages, whether requested due to the [use of mailAutoReply* attributes in a user's LDAP entry](#), or requested due to explicit use of the ["vacation" action](#) in a user's Sieve filter, as a form of notification message also. Such "vacation" messages may optionally be generated either in standard MDN format, or (for the benefit of recipients whose user agents lack good handling for standard MDN messages) as more "simple" messages consisting of a single text part. (Standard MDN format for a "vacation" message is requested by use of `mailAutoReplyMode: echo` in the user's LDAP entry, or use of the `:echo` argument to a `vacation` action in the user's Sieve script. The "simple" form of "vacation" message is requested by use of `mailAutoReplyMode: reply` in the user's LDAP entry, or use of the `:reply` argument to a `vacation` action in the user's Sieve script.)

Selective choice of language used in, and customization of the contents of, notification messages is possible, as discussed in [DSN language and customization](#) and [MDN language and customization](#) below. But keep in mind that any such language usage and customization must be within the guidelines of the overall notification message format, which in many cases is prescribed by the above-mentioned Internet standards.

47.3.1 DSN language and customization

As mentioned in the discussion of the sample DSN of [Example non-delivery DSN](#), there are a number of localizable, customizable files consulted when constructing DSNs; these files will be discussed further in [Customizing DSNs via the return files](#). By default, if no NOTIFICATION_LANGUAGE mapping table is configured, the `return_*. *` files located in the `IMTA_LANG:` directory will be used to generate all DSNs. However, separate sets of `return_*. *` files can be created and located in separate directories--for localization or site customization purposes. Indeed, the MTA is distributed with several sets of language-specific `return_*. *` files, and a basic NOTIFICATION_LANGUAGE mapping table (referenced in

the mappings file via file inclusion of the `mappings.locale` file) which selects among the language-specific directories based on language preference of original message senders. That is, the `NOTIFICATION_LANGUAGE` mapping table selects, based on any language preference of the original message sender, a directory in which to find an appropriate set of `return_*.txt` files for generating a DSN back to that sender.

Probes to the `NOTIFICATION_LANGUAGE` mapping table have the form:

```
dsn-type | source-channel | accept-language | return-address | 1st-recipient
```

In the probe, `dsn-type` can be a comma-separated list including any of `failed`, `bounced`, `timedout`, `delayed`, `deferred`, `delivered`, `read`, `relayed`, `expanded`, `capture`, `error`, or (as of MS 7.0u2) `journal`. The `accept-language` field will have any values (possibly a comma-separated list) found on (preferentially) an `Accept-Language:` header line in the original message (the message that the DSN will be reporting on), or if that header line is not present then from a `Preferred-language:` header line if present, or failing that an `X-Accept-Language:` header line. Note that valid language tag values are discussed in [RFC 3066 \(Tags for the Identification of Languages\)](#).

The pattern (right hand side) of a `NOTIFICATION_LANGUAGE` entry may set the `$I` flag to specify the directory (or as of MS 7.0 update 3, a list of comma-separated directories) from which the MTA should use `return_*.txt` files when constructing a DSN. It may also optionally set the `$T` flag to set an override `Subject:` header field value to use in the DSN. Note that the default `Subject:` field values for the various types of DSNs are shown in [Table of DSN types and their default Subject: field text](#). When both flags are set, the directory argument should be first (left-most), separated by a vertical bar character from the `Subject:` header field.

Note: If using `NOTIFICATION_LANGUAGE` to select an alternate directory of `return_*.txt` regardless of DSN type, then it is important to have a *complete* set of `return_*.txt` files present in the specified directory. If the MTA cannot locate a `return_*.txt` file of the appropriate type in the directory in which the MTA has been told to find such files, then the DSN will be constructed omitting that part of the usual structure and the resulting DSN will therefore look "odd" or "incomplete".

Note: When the MTA has been configured to send copies of notifications to the postmaster (see for instance the [sendpost](#) and [warnpost](#) channel options), those postmaster copies of the notification message back to the original sender are copies of the text sent back to the original sender, and in particular are in the language selected for the original sender. The postmaster's own personal language preferences, if any, are not relevant to these copies; the purpose of these copies being to be a *copy* of what the original sender was sent.

47.3.1.1 Customizing DSNs via the `return_*.txt` files

When the MTA needs to construct a notification message, the MTA will consult the [NOTIFICATION_LANGUAGE mapping table](#) to find a language-appropriate set of `return_*.txt` files. The MTA will then use that set of `return_*.txt` files to construct the notification message. The [return_header.opt](#) file and (optional) [return_option.opt](#) file are modifier files, potentially affecting the spectrum of notification messages. The [return_*.txt](#) files, in contrast, are template files for the different types of notification messages. Each such `return_*.txt` file is used nearly verbatim for its particular type(s) of notification messages, with possible substitutions as specified via `%` substitutions. (The [notary_quote](#) MTA option controls this meaning of the percent character, `%`, in notification message template files.) Note that in order to specify inserting a literal percent character in a `return_*.txt` file, the percent character must itself be quoted with another percent character, `%%`.

47.3.1.1.1 The sample header lines in DSNs: the `return_header.opt` file

The purpose of the `return_header.opt` file is to specify which message header lines to include to identify a message in the human-readable first part of DSNs. Typically it is desired to return only a few of the possible header lines a message might contain, only those most significant to the original message sender.

Prior to MS 7.0 update 2, the `return_header.opt` file was not language-specific: it always resided and was found in the `IMTA_LANG:` directory. All the other `return_*.opt` files may be language-specific, as the directory in which to locate them may be varied using the [NOTIFICATION_LANGUAGE mapping table](#). As of MS 7.0 update 2, the MTA will first look for a language-specific `return_header.opt` in the directory (or as of MS 7.0 update 3, list of directories) selected via `$I` in the `NOTIFICATION_LANGUAGE` or `DISPOSITION_LANGUAGE` mapping table, as relevant; but if the MTA does not find a language-specific such file, then it will fall back to looking in the `LANGDIR` directory instead. (As of JES MS 7.0, the former `IMTA_LANG` Tailor file option is deprecated and replaced by the `langdir` MTA option as far as locating on disk where files are located. But `IMTA_LANG` may still be used in, for instance, mapping tables: the MTA will translate occurrences of `IMTA_LANG` to the location specified by the `langdir` MTA option.)

As a concrete example, the `return_header.opt` file installed in the `IMTA_LANG:` directory is shown [here](#). The settings shown [here](#) mean that the `%H` substitution typically used in the `return_prefix.txt` file will cause insertion of solely the Message-Id:, Date:, From:, To:, and Subject: header lines of the original message into the first, human-readable portion of the DSN messages constructed by the MTA; all other header lines from the original message will be omitted. See the `imsimta test -header` utility for an example of application of this header trimming to a set of headers.

Sample distributed `return_header.opt` file

```
Message-Id: PRECEDENCE=3
Date: PRECEDENCE=4
From: PRECEDENCE=5
To: PRECEDENCE=6
Subject: PRECEDENCE=7
Others: MAXIMUM=-1
Defaults: MAXIMUM=-1
```

47.3.1.1.2 The required full set of DSN type-specific `return_*.txt` files

Within a language-specific directory as selected via the `NOTIFICATION_LANGUAGE` mapping table, each type of DSN should have its own `return_dsn-type.txt` file, to be used in constructing the text in the human-readable first part of the DSN, where the `dsn-type` in the file name is also the value in the `NOTIFICATION_LANGUAGE` probe. Each type of DSN also has a default Subject: header field value, shown in [Table of DSN types and their default Subject: field text](#), though these values may be overridden either via the `NOTIFICATION_LANGUAGE` mapping table as described in [DSN language and customization](#) or via the `SUBJECT` option in the `return_option.opt` file. [Table of DSN types and their default Subject: field text](#) lists the usage of each of these `return_dsn-type.txt` files.

Table 47.1 DSN types and their default Subject: field text

Type of DSN	Subject: field default text ²	Usage
-------------	--	-------

capture	Message Capture Copy	Used for the "capture" copy of a message captured due to Sieve script or LDAP attribute capture
bounced (security)	Delivery Notification: Potential security problem found	
bounced (manual)	Delivery Notification: Delivery has been manually aborted	The message was manually returned due to the postmaster using a utility such as <code>imsimta return</code> or the <code>imsimta qm</code> utility's <code>return</code> command to bounce the message; this is one case of "R" MTA message transaction log entries
timedout	Delivery Notification: Delivery has timed out and failed	Used when generating a bounce message due to a message exceeding the final <code>notices</code> value; this is one case of "R" MTA message transaction log entries
failed	Delivery Notification: Delivery has failed	Recipient encountered a permanent delivery failure (a rejection); this is one case of "R" MTA message transaction log entries
deferred	Delivery Notification: Delivery has been deferred	Used when a notification is generated that a message that had a "Deliver on first try" notification request set encountered a temporary failure on that first attempt
delayed	Delivery Notification: Delivery has been delayed	Used when a warning notification is generated for a message that remains in the MTA's queues as yet undelivered: that is, this is the text used for warning messages generated at the non-final <code>notices</code> values, so corresponding to "W" MTA message transaction log entries
delivered	Delivery Notification: Delivery has been successful	Used when generating a delivery receipt for a message that had a NOTARY delivery receipt request
relayed	Delivery Notification: Message successfully relayed	Used when a message that had a NOTARY delivery receipt request is relayed onward to a host that does not support NOTARY
expanded	Delivery Notification: Mailing list successfully expanded	Used when an address that had a delivery receipt notification request is expanded into a mailing list ; per NOTARY rules, such expansion is considered to be a successful "delivery" of the message (and the delivery receipt request does not get propagated/ carried through to the message copies addressed to actual list members)
error	Problem during delivery processing	Used, for instance, for error reports to a Sieve "owner" regarding Sieve syntax problems
journal ¹	Message Journal Copy	Used in messages captured with the Sieve capture action's :journal parameter

¹ New in MS 7.0u2.

² The Subject: field value can be overridden by setting a single value to be used for *all* DSNs via the SUBJECT option of `return_option.opt`. Alternatively, a more complex approach is to use the \$T flag in the `NOTIFICATION_LANGUAGE` mapping table to set Subject: field values.

As a concrete example, the distributed English language set of `return_*.txt` files will be presented here. The `prefix`, `bounced`, `capture`, `deferred`, `delayed`, `delivered`, `error`, `failed`, `forwarded`, `timedout`, and `suffix` files are the ones normally found in the directory located via the `langdir` MTA option, typically `IMTA_ROOT:config/locale/C`. The MTA is also distributed with sets of `return_*.txt` files in other, language-specific directories under `IMTA_ROOT:config/locale`. Note that regardless of what language is a site's own "preferred" language, notifications may be generated by the MTA in other languages, using the `return_*.txt` files from other language-specific directories, according to any expressed language preference of the *original* message sender (who will receive the DSN). That is, it is not a site's own language preference, but rather the language preference of the DSN recipient -- who is possibly a remote user---that matters most when it comes to DSN language!

Sample English language `return_prefix.txt` file

```
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
```

Sample English language `return_bounced.txt` file

```
Your message is being returned.  It was forced to return by the postmaster.

The recipient list for this message was:
%R
```

Sample English language `return_capture.txt` file

```
Attached message captured in accordance with site policy.
```

Sample English language `return_deferred.txt` file

```
This system has been unable to deliver your message to the
following recipients:
%R
```

Sample English language `return_delayed.txt` file

```
Your message has been enqueued and undeliverable for %C %u%s
to the following recipients:
%R
```

The mail system will continue to try to deliver your message for an additional %L %u%s.

Sample English language return_delivered.txt file

Your message has been successfully delivered to the following recipients:
%R

Sample English language return_error.txt file

Processing errors occurred during delivery:
%R

Delivery processing continued in spite of these errors.

Sample English language return_failed.txt file

Your message cannot be delivered to the following recipients:
%R

Sample English language return_forwarded.txt file

Your message has been successfully relayed to the recipients
%R

on a remote system that does not support the generation of successful delivery receipts. This does NOT mean that your message has actually been placed in the recipients' mailboxes; merely that it has passed through a part of the message transport infrastructure. In the event of a nondelivery you should expect to receive a nondelivery notification; in the event of successful delivery, however, you are unlikely to receive a positive confirmation of delivery.

Sample English language return_timedout.txt file

Your message is being returned; it has been enqueued and undeliverable for %C %u%s to the following recipients:
%R

Sample English language return_suffix.txt file---normally empty

(Note: the return_suffix.txt file is typically empty.)

47.3.1.1.3 The optional return_option.opt file

The return_option.opt file is the one optional file among the return_*. * files; it need not exist, as in its absence, a reasonable set of (English language) default values will be used.

(Indeed, the English language defaulting for the text controlled by this file is actually a bit more sophisticated than the handling available via the options in the file.) However, while (unlike the other `return_*. * files`) its existence is not *required*, it is normally desirable to have such a file in non-English, language-specific directories. The options available for `return_option.opt` are listed below. Note that it is critical that any CHARSET setting in the [return_prefix.txt file](#) be coordinated with the values used (the representation of text used) in `return_option.opt` option values: the CHARSET must match (except for SUBJECT which, if non-US-ASCII characters are to be presented, must be specified as an already RFC 2047 encoded value).

47.3.1.1.3.1 DAY (string), HOUR (string)

Specify the text string to use for %U and %u substitutions. When the MTA option `return_units=0` (units of days) is set, the string specified for DAY will be used; when the MTA option `return_units=1` (units of hours) is set, the string specified for HOUR will be used. So for instance, in a French language version of `return_option.opt`:

```
DAY=jour
HOUR=heure
```

Note that the English language default handling, if no DAY and HOUR options are set, is slightly more sophisticated than available with the option, in that the default English language handling includes a case distinction: it will substitute Day or Hour in place of %U, while substituting "day" or "hour" in place of %u. It is important that the actual values be specified in the CHARSET matching the CHARSET parameter configured for the Content-type: header line in the [return_prefix.txt file](#).

47.3.1.1.3.2 DIAGNOSTIC_CODE (string)

The English language default is:

```
DIAGNOSTIC_CODE= Diagnostic code:
```

or for example a French language setting could be:

```
DIAGNOSTIC_CODE= Code de diagnostic :
```

47.3.1.1.3.3 ORIGINAL_ADDRESS (string)

The English language default is:

```
ORIGINAL_ADDRESS= Original address:
```

Or for example a French language setting could be:

```
ORIGINAL_ADDRESS= Adresse d'origine :
```

47.3.1.1.3.4 RETURN_PERSONAL (RFC 2047-encoded string)

Specify the personal name (RFC 822 phrase) to use with the postmaster address. If specified, this will override the Postmaster personal name specified for the source channel via

`returnpersonal` as well as the global default specified via the `return_personal` MTA option.

47.3.1.1.3.5 SUBJECT (RFC 2047-encoded string)

Specify the value to use on the Subject: header line of DSNs, in RFC 2047 encoded form. So for instance in a French language version of `return_option.opt`:

```
SUBJECT==?iso-8859-1?Q?Notification_de_l=27=E9tat_de_remise?=
```

Note that the English language handling, if SUBJECT is not specified, is slightly more sophisticated than available via this option as different values will be used for different types of DSNs; see [DSN types and their default Subject: field text](#). It is also possible to specify use of different Subject: values via the \$T flag of the [NOTIFICATION_LANGUAGE mapping table](#).

47.3.1.1.3.6 REASON (string)

The English language default is:

```
REASON= Reason:
```

or for example a French language setting could be:

```
REASON= Raison :
```

47.3.1.1.3.7 RECIPIENT_ADDRESS (string)

The English language default is effectively:

```
RECIPIENT_ADDRESS= Recipient address:
```

or for example a French language setting could be:

```
RECIPIENT_ADDRESS= Adresse du destinataire :
```

47.3.1.1.3.8 REMOTE_SYSTEM (string)

The English language default is:

```
REMOTE_SYSTEM= Remote system:
```

or for example a French language setting could be:

```
REMOTE_SYSTEM= Systeme distant :
```

47.3.1.1.3.9 x.y.z (string)

For any possible SMTP enhanced status code that might be reported in a notification, it is possible to configure additional, alternate text to include in the human-readable portion of

the notification. Such text might be chosen to perhaps "better explain" (or at least provide a description in an alternate language) the corresponding enhanced status code. For instance:

```
5.1.0=Il y avait un erreur indefini avec l'adresse du destinaire.
5.1.1=La boite aux lettres a l'adresse specifiee n'existe pas.
5.1.2=Le systeme du destination specifiee dan l'adresse n'existe ou est incapable d'accepter la poste.
```

etc. See [RFC 1893](#) for definitions of the standard meanings of enhanced status codes. Such explanation text, if specified, will be output after the ORIGINAL_ADDRESS information and before the REASON information as part of the %R recipient information substitution.

47.3.1.1.3.10 Option usage

So in particular, note that `return_option.opt` can re-define the meanings of some of the various % substitutions available in the `return_*.txt` files. The available substitution sequences, along with any `return_option.opt` options controlling their text effect, are shown in [return_*.txt file substitution sequences](#).

Table 47.2 `return_*.txt` file substitution sequences

Sequence	return_option.opt option	Default value	Meaning
%%		%	Substitute a literal % character.
%B			Substitute a boundary marker.
%C			Substitute the length of time (in days or hours---see the return_units MTA option and the %u and %U substitutions) the message has been queued; that is, the length of time during which the MTA has been trying to deliver a not-yet-delivered message.
%F			Substitute the length of the time the message may remain in the queue prior to the MTA bouncing it; that is, the final <code>backoff</code> keyword value minus the %C time-queued-so-far.
%H			Substitute those header lines specified in the <code>return_header.opt</code> file. See the notary_decode MTA option for discussion of decoding of material encoded due to use of non-US-ASCII characters.
%I			Substitute the length of time until the next notice; the next <code>notices</code> value minus the current %C time-queued-so-far.
%L			Substitute the length of time remaining until the MTA will bounce (return) a not-yet-delivered message; the length of time until the final <code>backoff</code> keyword value. See the return_units MTA option for whether this length of time is in units of days or hours; and see the %u and %U unit name substitutions.
%N			Substitute the length of time corresponding to the next <code>notices</code> value; this is the total amount of time between enqueue and that notice being due for generation, not the remaining time (for which see instead %I).
%O			(New in MS 6.2) Treat the % character as having no special meaning in the remainder of this <code>return_*.txt</code> file. That is, disable all % substitution sequence interpretation in the remainder of this template file.
%R	<i>see below</i>	Recipient address: <i>address</i> Original address: <i>orcpt-value</i> Reason: <i>reason</i> Diagnostic code: <i>SMTP-error</i> Remote system: <i>name-and-details</i>	The %R substitution causes output of a whole set of recipient-specific information, for each relevant envelope recipient, as a set of lines in <i>customizable-field-name field-value</i> format. These are intended as human-readable analogues (and in particular customizable and localizable versions for use in the human-readable portion of the DSN) of some of the standardized

			fields that RFC 3464 defined for use in the machine-readable portion of the DSN. The <i>customizable-field-name</i> text may be customized using a number of <code>return_option.opt</code> options.
%R	RECIPIENT_ADDRESS	Recipient address:	The setting of includefinal , suppressfinal , or useintermediate on the current source channel (the channel generating the notification) can affect what form of the recipient address is presented
%R	ORIGINAL_ADDRESS	Original address:	
%R	x.y.z	<i>MTA-error-text-for-x.y.z-status</i>	Substitute text explaining the "meaning" of the extended SMTP status <i>x.y.z</i> ---typically this would be alternate language text intended to be more comprehensible than the English language SMTP error text.
%R	REASON	Reason:	
%R	DIAGNOSTIC_CODE	Diagnostic code:	
%R	REMOTE_SYSTEM	Remote system:	
%s		s	Substitute a literal "s" character if the previously substituted numeric value was not equal to one; typically used after a %u or %U substitution.
%S		S	Substitute a literal "S" character if the previously substituted numeric value was not equal to one; typically used after a %u or %U substitution.
%u	DAY	day	
%u	HOUR	hour	
%U	DAY	Day	
%U	HOUR	Hour	

47.3.2 MDN language and customization

There are a number of localizable, customizable files the MTA consults when constructing MDNs; these files will be discussed further in [Customizing MDNs via the disposition files](#). By default, if no `DISPOSITION_LANGUAGE` mapping table is configured, the `disposition_*. *` files located in the `langdir` directory will be used to generate all MDNs. However, separate sets of `disposition_*. *` files can be created and located in separate directories---for localization or site customization purposes. Indeed, the MTA is distributed with several sets of language-specific `disposition_*. *` files, and a basic `DISPOSITION_LANGUAGE` mapping table (referenced in the mappings file via file inclusion of the `mappings.locale` file) which selects among the language-specific directories based on language preference of original message senders. That is, the `DISPOSITION_LANGUAGE` mapping table selects, based on any language preference of the original message sender, a directory in which to find an appropriate set of `disposition_*. *` files for generating an MDN back to that sender.

Probes to the `DISPOSITION_LANGUAGE` mapping table have the form:

```
mdn-type | modifier | source-channel | accept-language | return-address | recipient
```

where `mdn-type` can be any of `displayed`, `dispatched`, `processed`, `deleted`, `denied`, or `failed`, and where `modifier` can be a comma-separated list including any of `error`, `warning`, `superseded`, or `expired`. The `accept-language` field will have any values (possibly a comma-separated list) found on (preferentially) an `Accept-Language:` header line, or if that header line is not present then from a `Preferred-language:` header line if present, or failing that an `X-Accept-Language:` header line. Note that valid language tag values are discussed in [RFC 3066 \(Tags for the Identification of Languages\)](#).

The pattern (right hand side) of a `DISPOSITION_LANGUAGE` entry may set the `$I` flag to specify the directory from which the MTA should use `disposition_*. *` files when

constructing an MDN, and optionally following a vertical bar character a destination charset. The pattern may also optionally use the `$T` flag to set the Subject: field value to use when constructing this MDN; the Subject: field follows yet another vertical bar character. (Note that the `$T` Subject: will only be used if there is neither a specific Subject: field for this type of MDN (such as a Sieve "vacation :subject" might specify), nor a SUBJECT option set in the `disposition_option.opt` file. That is, the `$T` specified Subject: field value is of low precedence compared to the other ways of setting the Subject: field value.) The pattern may also optionally set the names of the files to use when constructing this MDN instead of the normal `disposition_prefix.txt`, `disposition_mdn-type.txt`, and `disposition_suffix.txt` files; these are specified following the directory specification but prior to the first vertical bar character, with comma separators. All together, the syntax of the pattern is:

```
$I$Tlangdir,option,prefix,mdn-type-file,suffix|dest-charset|subject
```

See [Sample NOTIFICATION_LANGUAGE](#) and [DISPOSITION_LANGUAGE](#) mapping tables for an example DISPOSITION_LANGUAGE mapping table.

Note that as of MS 7.0.5, all MDNs generated by the MTA include an Auto-Submitted: header line, per the recommendation of [RFC 3834 \(Recommendations for Automatic Responses to Electronic Mail\)](#) -- in prior versions only certain forms of MDNs contained such a header line -- and the value placed on the Auto-Submitted: header line has been updated to include the additional information suggested in [RFC 5436 \(Sieve Notification Mechanism: mailto\)](#) including additional values (such as auto-notified) and the owner-email parameter.

47.3.2.1 Customizing MDNs via the `disposition_*.txt` files

When the MTA needs to construct an MDN, the MTA will consult the [DISPOSITION_LANGUAGE](#) mapping table to find a language-appropriate set of `disposition_*.txt` files, and optionally some override files and values. The MTA will then use that set of `disposition_*.txt` files, plus as relevant the `return_option.opt` and `return_header.opt` files discussed in [Customizing DSNs via the return files](#), to construct the MDN. The `return_option.opt` file, `return_header.opt` file, and (optional) `disposition_option.opt` files are modifier files, potentially affecting the spectrum of MDN messages. The `disposition_*.txt` files, in contrast, are template files for the different types of MDNs. The `disposition_prefix.txt` and `disposition_suffix.txt` template files are used to "wrap" the MDN-type-specific text; each such `disposition_MDN-type.txt` file is used nearly verbatim for its particular type(s) of MDN, with possible substitutions as specified via `%` substitutions (defined from `return_option.opt`). (The [notary_quote](#) MTA option controls this meaning of the percent character, `%`, in MDN template files.) Note that in order to specify inserting a literal percent character in a `disposition_*.txt` file, the percent character must itself be quoted with another percent character, `%%`.

47.3.2.1.1 The `disposition_*.txt` files

The set of MDN `disposition_*.txt` template files is:

```
disposition_prefix.txt
disposition_deleted.txt
disposition_denied.txt
disposition_dispatched.txt
```

```
disposition_displayed.txt
disposition_failed.txt
disposition_processed.txt
disposition_suffix.txt
```

But note that currently, of these MDN-type files, only `disposition_deleted.txt` (an [old-style Sieve "reject" action](#)), and `disposition_dispatched.txt` ([Sieve "notify" action](#)), are routinely used by the MTA as those (and some forms of vacation messages) are the types of MDNs that the MTA routinely generates---and vacation messages have their own distinct mechanisms for generating their text. The `disposition_prefix.txt` and `disposition_suffix.txt` files are also used, to prefix and suffix, respectively, the context placed in an MDN.

Each type of MDN has a default Subject: header field value, shown in [MDN types and their default Subject: field text](#), though these values may be overridden either via the [DISPOSITION_LANGUAGE mapping table](#) or via the SUBJECT option in the `disposition_option.opt` file. Furthermore, some types of MDNs have their own, higher-precedence way, of specifying the Subject: header field value---as for instance the `"vacation :subject"` parameter. [MDN types and their default Subject: field text](#) lists the usage of each of these `disposition_mdn-type.txt` files.

Table 47.3 MDN types and their default Subject: field text

Type of MDN	Subject: field default text ¹	Usage
deleted	Message has been deleted	Old-style Sieve "reject" action
denied	Message has been denied	
displayed	Message has been displayed	Would be used for read receipts, if the MTA generated read receipts (which it doesn't--- read receipts are up to user agents to generate)
dispatched	Message has been dispatched	Sieve "notify" action
failed	Message has failed	
processed	Message has been processed	<code>"vacation :echo"</code> messages (or <code>mailAutoReplyMode: echo</code> messages); note that vacation messages may set their own Subject: field value

¹ The Subject: field value can be overridden by setting a single value to be used for *all* MDNs via the SUBJECT option of `disposition_option.opt`. Alternatively, a more complex approach is to use the \$T flag in the [DISPOSITION_LANGUAGE mapping table](#) to set Subject: field values. Certain types of MDNs have their own, more explicit, methods for setting the Subject: field value.

As a concrete example, a few of the distributed English language set of `disposition_*.txt` files will be presented here. Shown in [Sample English language disposition_prefix.txt file](#), [Sample English language disposition_deleted.txt file](#), [Sample English language disposition_dispatched.txt file](#), and [Sample English language disposition_suffix.txt file](#) these are some of the files typically found in the directory

located via the `langdir` MTA option, `IMTA_ROOT:config/locale/C/`. The MTA is also distributed with sets of `disposition_*.txt` files in other, language-specific directories under `IMTA_ROOT:config/locale`. Note that regardless of what language is a site's own "preferred" language, notifications such as MDNs may be generated by the MTA in other languages, using the `disposition_*.txt` files from other language-specific directories, according to any expressed language preference of the *original* message sender (who will receive the MDN). That is, it is not a site's own language preference, but rather the language preference of the MDN recipient -- who is possibly a remote user -- that matters most when it comes to MDN language!

Sample English language `disposition_prefix.txt` file

```
Content-type: text/plain; charset=us-ascii
Content-language: EN-US
```

```
This disposition report relates to a message you sent with the following header
fields:
%H
```

Sample English language `disposition_deleted.txt` file

```
Your message was refused by %R and has been deleted.
The reason given for the deletion was the following:
```

Sample English language `disposition_dispatched.txt` file

```
Your message has been sent somewhere in some manner (e.g., printed,
faxed, forwarded) without necessarily having been read by the recipient
%R.
```

Sample English language `disposition_suffix.txt` file---normally empty

(Note: the `disposition_suffix.txt` file is typically empty.)

47.3.2.1.2 The `disposition_option.opt` file

The `disposition_option.opt` file is optional; in its absence, a reasonable set of (English language) default values will be used. (Indeed, the English language defaulting for the text controlled by this file is actually a bit more sophisticated than the handling available via the options in the file.) However, while its existence is not *required*, it is normally desirable to have such a file in non-English, language-specific directories. The `disposition_option.opt` file supports the following options:

47.3.2.1.2.1 SUBJECT

The SUBJECT option sets a default field value for the Subject: header line. The precedence is that any Subject: field value specific to the type of MDN -- for instance, a subject specified via a Sieve "vacation" action---takes precedence over the setting of the SUBJECT option. But the SUBJECT option in turn, when set, takes precedence over any Subject: field value set via the \$T flag in the DISPOSITION_LANGUAGE mapping table, which in turn takes precedence over the default text shown in MDN types and their default Subject: field text.

47.3.2.1.2.2 RETURN_PERSONAL

When an MDN with a From: header address of the [postmaster address](#) is being generated, (which note is not the case for all types of MDNs), then the RETURN_PERSONAL option sets the so-called "personal name" (technically, the RFC 822 "phrase") to use with the postmaster address.

47.3.2.1.2.3 TEXT_CHARSET

The TEXT_CHARSET option specifies the charset for the disposition_*.txt files.

47.3.3 NOTIFICATION_LANGUAGE and DISPOSITION_LANGUAGE sample mapping tables

The normal configuration of the MTA, established by initial configuration, includes basic NOTIFICATION_LANGUAGE and DISPOSITION_LANGUAGE mapping tables, as well as subsidiary mapping tables. (In legacy configuration, these mappings were included into the main MTA mappings file via a reference to the distributed file mappings.locale; these language-related mappings were stored in mappings.locale for MTA administrator convenience, as they are rather large mappings, complicated to read, and seldom changed by MTA administrators.)

In Unified Configuration, these mappings appear as:

```
msconfig> show mapping:NOTIFICATION_LANGUAGE
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|en*|*|* $IIMTA_TABLE:locale/C/,IMTA_LIB:locale/C/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|de*|*|* $IIMTA_TABLE:locale/de/,IMTA_LIB:locale/de/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|es*|*|* $IIMTA_TABLE:locale/es/,IMTA_LIB:locale/es/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|fr*|*|* $IIMTA_TABLE:locale/fr/,IMTA_LIB:locale/fr/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|ja*|*|* $IIMTA_TABLE:locale/ja/,IMTA_LIB:locale/ja/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|zh-TW*|*|* $IIMTA_TABLE:locale/zh_TW/,IMTA_LIB:locale/zh_TW/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|zh*|*|* $IIMTA_TABLE:locale/zh/,IMTA_LIB:locale/zh/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|ko*|*|* $IIMTA_TABLE:locale/ko/,IMTA_LIB:locale/ko/
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|$_*,$T**|*|* $R$0|$1|$4|$5|$6
role.mapping:NOTIFICATION_LANGUAGE.rule = *|*|*|*|* $CS|LDAP_USERS_LANGUAGE;$3@$4|$E
msconfig> show mapping:LDAP_USERS_LANGUAGE
role.mapping:LDAP_USERS_LANGUAGE.rule = *|*|*|*|* $C|DC|$0@$1|$|DOMAIN_DC|$1|
role.mapping:LDAP_USERS_LANGUAGE.rule = |DC|*|*|* $C|BDN|$0@$1|$2|$1ldap:///S2,o=internet?inetDomainBaseDN?sub?(objectClass=inetDomain){
role.mapping:LDAP_USERS_LANGUAGE.rule = |DC|*|*|* $C|BDN|$0@$1|$2|$1ldap:///S2,o=internet?aliasedObjectName?sub?(objectClass=inetDomainAlias){
role.mapping:LDAP_USERS_LANGUAGE.rule =
|BDN|*|*|* $C|LANG|$1ldap:///S2?preferredLanguage?sub?{(mail=$0$_) (mailAlternateAddress=$0$_) (mailEquivalentAddress=$0$_)}|
role.mapping:LDAP_USERS_LANGUAGE.rule =
|BDN|*|*|* $C|LANG|$1ldap:///S1,o=internet?preferredLanguage?sub?{(objectClass=inetDomain) (objectClass=inetDomainAlias)}|
role.mapping:LDAP_USERS_LANGUAGE.rule = |LANG|* $CIMTA_TABLE:locale/$|LANGUAGE_LOCALES;$0|/,IMTA_LIB:locale/$|LANGUAGE_LOCALES;$0|/$$ISY$E
msconfig> show mapping:LANGUAGE_LOCALES
role.mapping:LANGUAGE_LOCALES.rule = en C$Y
role.mapping:LANGUAGE_LOCALES.rule = de de$Y
role.mapping:LANGUAGE_LOCALES.rule = es es$Y
role.mapping:LANGUAGE_LOCALES.rule = fr fr$Y
role.mapping:LANGUAGE_LOCALES.rule = ja ja$Y
role.mapping:LANGUAGE_LOCALES.rule = zh-TW zh_TW$Y
role.mapping:LANGUAGE_LOCALES.rule = zh zh$Y
role.mapping:LANGUAGE_LOCALES.rule = ko ko$Y
role.mapping:LANGUAGE_LOCALES.rule = * $N
msconfig> show mapping:DISPOSITION_LANGUAGE
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|en*|*|* $IIMTA_TABLE:locale/C/,IMTA_LIB:locale/C/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|de*|*|* $IIMTA_TABLE:locale/de/,IMTA_LIB:locale/de/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|es*|*|* $IIMTA_TABLE:locale/es/,IMTA_LIB:locale/es/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|fr*|*|* $IIMTA_TABLE:locale/fr/,IMTA_LIB:locale/fr/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|ja*|*|* $IIMTA_TABLE:locale/ja/,IMTA_LIB:locale/ja/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|zh-TW*|*|* $IIMTA_TABLE:locale/zh_TW/,IMTA_LIB:locale/zh_TW/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|zh*|*|* $IIMTA_TABLE:locale/zh/,IMTA_LIB:locale/zh/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|ko*|*|* $IIMTA_TABLE:locale/ko/,IMTA_LIB:locale/ko/|UTF-8
role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|$_*,$T**|*|* $R$0|$1|$2|$5|$6|$7
```

```

role.mapping:DISPOSITION_LANGUAGE.rule = *|*|*|*|*|*|* $C$|LDAP_USERS2_LANGUAGE;$4@$5|$E
msconfig> show mapping:LDAP_USERS2_LANGUAGE
role.mapping:LDAP_USERS2_LANGUAGE.rule = *@* $C|BDN|$0@$1|$1,$1,_base_dn_{
role.mapping:LDAP_USERS2_LANGUAGE.rule =
|BDN|*|* $C|LANG|$1ldap://$1?preferredLanguage=sub?{(mail=$=$0$_) (mailAlternateAddress=$=$0$_) (mailEquivalentAddress=$=$0$_) }{
role.mapping:LDAP_USERS2_LANGUAGE.rule = |BDN|*|*|* $C|LANG|$1,$1,preferredLanguage{
role.mapping:LDAP_USERS2_LANGUAGE.rule =
|LANG|* $C|MTA_TABLE:locale/$|LANGUAGE_LOCALES;$0|/,IMTA_LIB:locale/$|LANGUAGE_LOCALES;$0|/$$I|UTF-8$$Y$E

```

Note: Some output lines have been wrapped for clarity.

47.4 Notification message routing

When the MTA needs to generate a notification message, its default is to enqueue the notification message to the [process channel](#). The new notification message will be enqueued to the process channel from whatever channel encountered the condition that required the notification message to be generated. For instance, a `tcp_local` channel when attempting to send a message out to the Internet (that is, `tcp_local` operating as an SMTP client) may need to generate a notification message if a message it is attempting to deliver encounters a permanent rejection error from a remote SMTP server. Or the `tcp_local` channel's SMTP server may need to generate a notification message if an incoming message that it is attempting to process is addressed to a local recipient who has a Sieve filter containing a syntax error; in this case while the original message gets delivered as "normal", the `tcp_local` channel also needs to generate a notification message to the Sieve filter owner letting them know that their Sieve filter has a syntax error. The process channel will then send the notification message onwards to an appropriate destination channel. (This is in fact the main purpose of the process channel, namely to handle notification messages: accept them being generated by other channels, and then properly enqueue them onwards to their destinations.)

Optionally, the MTA may be configured, on a per-channel basis, to enqueue notification messages to alternate `process_*` channels via the [notificationchannel](#) and [dispositionchannel](#) channel options. This may be useful when a site processes a large number of notification messages. Splitting up notification message traffic into different `process_*` channels, and then optionally using source-channel-specific rewrite rules (or the [CONVERSIONS mapping table](#)) to route the messages coming from different `process_*` channels out through different destination channels, may aid in managing large volumes of notification messages. (In particular, this may be of interest with special purpose channels that are subject to unusually high volumes of notification messages, or when dealing with rejections of spam messages that had forged From: addresses claiming to come from unresponsive rather than clearly nonexistent domains.)

For instance, suppose one adds new, special channel definitions that are variants of the usual `tcp_local` and `process` channels, so along the lines of:

```

tcp_local_dsn_out smtp mx single_sys subdirs 20 maxjobs 7 pool SMTP_POOL \
  loopcheck
tcp-dsn-out-daemon

process_dsn_out
process-dsn-out-daemon

```

then adds `notificationchannel process_dsn_out` to the regular `tcp_intranet` channel definition:

```
tcp_intranet ...usual-keywords... notificationchannel process_dsn_out
tcp_intranet-daemon
```

and configures [special routing](#) by using

CONVERSIONS

```
IN-CHAN=process_dsn_out;OUT-CHAN=tcp_local;CONVERT \
Yes,Channel=tcp_local_dsn_out
```

This will result in notifications generated by the `tcp_intranet` channel regarding messages originally from Internet senders (notifications back to Internet senders regarding the Internet senders' messages originally addressed to recipients on other internal hosts) to be routed out the special `tcp_local_dsn_out` channel, rather than out the usual `tcp_local` channel. This can then allow for special handling, or special tuning of the handling, of such messages.

Also, the new-in-6.3-1.04 `$~` flag in the [FROM_ACCESS mapping table](#) provides, among other things, another way to do special routing of incoming notification messages (messages with empty envelope From). That is, this provides a way to associate a special source channel (and hence do special destination channel routing, or other special source-channel based handling) for notification messages generated external to this MTA. *E.g.*, with new, special channel definitions that are variants of the usual `tcp_local` and `tcp_lmtpcs` channels, so along the lines of:

```
tcp_local_dsn_in smtp mx single_sys remotehost inner switchchannel \
subdirs 20 maxjobs 7 pool SMTP_POOL maytlserver maysaslserver \
sasls witchchannel tcp_auth missingrecipientpolicy 0 loopcheck
tcp-dsn-in-daemon
```

```
tcp_lmtpcs_dsn_out defragment lmtp multigate connectcanonical \
fileinto @$40:$U+$S@$D port 225 nomx single_sys subdirs 20 maxjobs 7 \
pool SMTP_POOL dequeue_removertime
lmtpcs-dsn-daemon
```

then use mappings along the lines of:

FROM_ACCESS

```
TCP|*|SMTP*|MAIL|tcp_local||* $Y$~tcp_local_dsn_in
```

CONVERSIONS

```
IN-CHAN=tcp_local_dsn_in;OUT-CHAN=tcp_lmtpcs;CONVERT \
Yes,Channel=tcp_lmtpcs_dsn_out
```

to cause notifications coming in from the Internet destined for LMTP recipients to get routed out the special `tcp_lmtpcs_dsn_out` delivery channel, rather than out the usual `tcp_lmtpcs` delivery channel.

47.5 Bounces of spam messages

Joe-job or *blow back* spam are terms for the following occurrence. When spam (unsolicited bulk e-mail) is sent with a forged From: address pointing to one of your users (or at least an address putatively in your domain), if that spam is then rejected (bounced) by some intended spam recipient, the resulting notification messages (the bounces of the original spam messages) come back to *your* user, or at least your host! When one or more of your users' From: address gets forged on spam messages, this can then lead to a large volume, a "storm", of notification messages incoming to your users/hosts. One technique that can be helpful in such cases is to "isolate" incoming notification messages, as discussed in [Notification message routing](#), and then perhaps do such things as "slow down" delivery of such notifications relative to other messages (for instance, reducing the `maxjobs` for the special delivery channel) and/or perform specially rigorous scanning of such incoming notifications and, when appropriate, discard them rather than deliver them.

Another issue that can occur with notifications and spam messages is when spam messages come in to your host with forged invalid (but not immediately obvious as such) From: addresses. If the spam messages are bounced, this can lead to a large burden of notification messages (regarding the spam) which your MTA is attempting to send (deliver) to what are in fact undeliverable recipient addresses (those invalid forged purported original sender addresses). When forged From: addresses on the original spam point to domains that will result in merely temporary errors upon delivery attempts (of the notifications concerning the bounces of the original spam), your MTA's outgoing message channels can get burdened with these large numbers of notifications that will never be deliverable (but meantime are cluttering up the outbound delivery channels). One approach to deal with this is to discard (rather than bounce) such spam; however, some sites for legal or other reasons can not use such a discard approach. In such cases, use of a `notificationchannel` and then a source-channel-specific rewrite rule (or [CONVERSIONS mapping table](#) entry) to route messages coming from the special `notificationchannel` out a special outbound channel in turn, can prevent such messages from unduly burdening your MTA and unduly interfering with other message processing. See [Notification message routing](#) for an example of such configuration.

47.6 Notification message logging

[MTA message transaction log entries](#) showing an empty envelope From address are the indication of a notification message, whether generated externally and merely passing through the MTA, or whether generated by the MTA itself.

For notification messages generated by the MTA itself, note that notification messages are generated through the [process channel](#). Indeed, typically the generation of notification messages is the sole function of the process channel. So MTA message transaction log entries showing a message being enqueued to the process channel typically correspond exactly to the generation of a new notification message.

If the MTA option `log_message_id` is enabled, then when the MTA generates a notification message it will include both the original message-id and the message-id of the new, notification message in the message transaction log entry showing the notification message initial generation (that is, in the entry showing the notification message being enqueued to the process channel). Thus this allows correlating the original message with a corresponding notification message. Note that the presence of two message-id's in the message-id field is, also, a definite indication of MTA generation of a notification message.

Enabling the `log_process` MTA option is also helpful when investigating notification message generation, as it will show the same process enqueueing the newly generated notification message, and then recording whatever was the appropriate action on the original

message, *e.g.*, an "R" record, a "W" record, an "E" record (as in the case of a notification due to a Sieve syntax error report, or message capture notification), *etc.* A point to emphasize here is that the notification message generation, that is, the enqueue to the process channel, occurs and hence is recorded before the completion of the operation on the original message, hence before the recording of what occurred to the original message.

47.7 Message size limits and notification messages

Administrative limits on message size can have interactions with notification messages.

First, note that it is very important to ensure that users can receive notification messages when messages they send can not be delivered. And note that bounce messages (non-delivery notifications) will sometimes include the contents of the originally sent message; so a non-delivery notification may be larger than the originally sent message. So when imposing administrative limits on message size, it is wise practice to make the limit on the size of message that may be received somewhat larger than the limit on the size of message that may be sent.

New behavior in JES MS 6.3-0.15 is that during [address reversal](#) of the envelope return address (envelope From address) of a message, the MTA fetches the block limit associated with that envelope return address and will set the NOTARY (RFCs 1891-1894) flag RET=HDRS (return only message header lines, not message content) on the message if no explicit return policy was already specified and the message size exceeds the block limit. This reduces cases where non-delivery reports regarding large messages are undeliverable themselves due to size.

The MTA options [bounce_block_limit](#) and [content_return_block_limit](#), channel options such as [blocklimit](#), and user LDAP attributes such as `mailMsgMaxBlocks` (or more precisely, whatever attribute is named by the [ldap_blocklimit](#) MTA option), can all affect what (whether and how much of an original message) gets included in notification messages.

47.8 Postmaster addresses

There are two sides to postmaster addresses: the postmaster addresses for which your site accepts mail, and the postmaster addresses your site emits as the From: address on [notification messages your site generates](#).

Domain names for SMTP servers visible on the Internet are required to be able to accept mail addressed to `postmaster@domain`. This means that any host with an Internet-facing SMTP server, and any visible-on-the-Internet hosted domain names, are required to support a corresponding postmaster mailbox. And indeed, it is normal and strongly recommended to have a postmaster mailbox for each and every e-mail domain name (including any purely internal e-mail domain names) you support. The MTA itself will send warning messages and, (depending upon version) may default to [sending copies of users' bounce messages to the postmaster](#).

It is critical that the postmaster address be a valid address for receiving mail! Hosting the postmaster mailbox directly on the Internet-facing SMTP server system may reduce the potential for problems arising in forwarding a message onwards. However, in a multi-tier deployment it is possible that you will not want to have to have someone log on regularly to the Internet-facing SMTP server system to check postmaster mail. If you do wish to direct

postmaster mail to a different system, be sure to ensure that the connection between the e-mail Internet-facing system and that other system is a very reliable connection; and be prepared that if something happens to break that connection, you will want to immediately change the postmaster address on the Internet-facing SMTP server system to some other functioning address or be prepared for the potential for serious e-mail problems. (Bouncing postmaster mail is not pretty.) See also the [aliaspostmaster channel option](#) which can sometimes simplify consolidation of postmaster addresses.

The special postmaster local-part is defined (see [RFC 822](#)) to be case-insensitive even if local-parts in general are allowed to be case-sensitive, and the MTA has special code to treat postmaster case-insensitively even if it has been configured (see the [alias_case MTA option](#)) to allow other local-parts to be case-sensitive.

Normal MTA configuration includes at least one postmaster address as an alias (whether in the [alias file](#), or as an [alias in Unified Configuration](#)).

As regards what postmaster address(es) your site emits on the notification messages your site generates, the MTA has a variety of configuration controls: the [return_address](#) and [return_personal](#) MTA options for MTA-wide defaults, the [returnaddress](#) and [returnpersonal](#) channel options for channel-specific settings, and the `mailDomainReportAddress` domain-level LDAP attribute (more precisely, the LDAP attribute named by the [ldap_domain_attr_report_address](#) MTA option) for a domain-specific setting, as well as potential language-specific override of the Postmaster personal name via the `RETURN_PERSONAL` option in language-specific [return_option.opt files](#), or [FROM_ACCESS mapping table](#) overrides of the Postmaster address via `$ (or $)` flags. The MTA defaults for `return_address` and `return_personal` mean that the MTA defaults to emitting as Postmaster address `postmaster@local-host` where `local-host` is the [official_host_name](#) on the L channel, and defaults to using as Postmaster personal name "Internet Mail Delivery".

Certain non-MTA options also affect the postmaster address emitted in cases of postmaster messages generated by other (non-MTA) components of Messaging Server; see also the [noticercpt](#) and [noticesender](#) Alarm options.

Chapter 48 Message tracking and recall

48.1 Message tracking and recall setup and configuration	48-1
48.1.1 Memcache server setup	48-1
48.1.2 MTA Channel Setup	48-2
48.1.3 MTQP server setup	48-3

A general message tracking and recall facility has been implemented. The tracking aspect of this facility conforms to [RFC 3885](#), [RFC 3886](#), and [RFC 3887](#). The recall aspect is implemented as a tracking extension. Additionally, some tracking extensions have been implemented to make it possible for a tracking client to track and recall messages submitted by a non-tracking client.

In order to enable message tracking and recall, a [memcache server](#) must be operating, and the [tracking_mode](#) MTA option must be set to 1 to enable use of memcache for message tracking purposes. Which messages get tracking information stored is controlled by various [message tracking channel options](#).

48.1 Message tracking and recall setup and configuration

There are three parts to setting up message tracking and recall capabilities:

1. Setting up a shared memcached server (or other software that implements the memcache protocol in a compatible fashion) to store tracking/recall information.
2. Configuring all the MTAs in the deployment to enable tracking and recall.
3. Setting up [Message Tracking and Query Protocol \(MTQP\) servers](#) on every MTA and possibly additional hosts.

48.1.1 Memcache server setup

Setting up memcached or compatible server is beyond the scope of this document. That said, note that memcached setup is very simple: Configuration consists of a small number of options on the command line. Indeed, it's often possible to simply say:

```
memcached -l <ip-address>
```

Where <ip-address> is the IP address where memcached accepts connections.

Once a server implementing the memcache protocol has been set up, every MTA in the deployment have to be configured to access it. This is done with the [memcache_host](#) MTA option; there is no tracking-specific host setting at the present time. The [memcache_port](#) MTA option must be explicitly set if the server is listening on a port other than 11211. It may be appropriate to set the [memcache_expire](#) MTA option to an appropriate value as well.

Tracking/recall is enabled within the MTA by setting the [tracking_mode](#) MTA option to 1. Logging tracking identifiers is good idea; this is controlled by the [log_tracking](#) MTA option.

Taken together, and assuming the memcache server is listening on the default port, the MTA option settings should look something like this:

```
msconfig> show memcache_host
role.mta.memcache_host memcache-server-ip
msconfig> show tracking_mode
role.mta.tracking_mode 1
msconfig> show log_tracking
role.mta.log_tracking 1
```

or in legacy configuration:

```
MEMCACHE_HOST=<memcache-server-ip>
TRACKING_MODE=1
LOG_TRACKING=1
```

48.1.2 MTA Channel Setup

The next step is to tell the tracking subsystem the semantics of the various channels in the deployment using the `trackinginternal`, `trackingrelayed`, and `trackingdelivered`. The general rules are:

1. Internal processing channels such as `process`, `reprocess`, `conversion`, and `defragment` require no special decoration.
2. Channels that perform final delivery must be marked with the `trackingdelivered` channel option. This includes not only `ims-ms` and `tcp_lmtpcs` channels, but also the `bitbucket` and `filter_discard` channels.
3. Any channel that relays messages to systems outside the deployment must be marked with the `trackingrelayed` channel option. Of course this includes the `tcp_local` channel, but would also include, say a special `tcp_aol` channel set up to handle mail to the `aol.com` domain.
4. Finally, any channel that relays message to other MTAs inside the deployment where tracking is enabled. This would typically be something like a `tcp_intranet` channel.

Note that for tracking and recall to work messages relayed to external systems **MUST** be handled by a different channel than message relayed to other MTAs inside the deployment. Meeting this requirement may require additional configuration changes.

The MTRK SMTP extension defined in [RFC 3885](#) is used to transfer tracking/recall information from one MTA to another. Use of this extension **MUST** be enabled on SMTP connections between MTAs inside the deployment. It also must be enabled on SUBMIT channels used by tracking/recall-enabled clients.

Since MTRK is an SMTP extension, its use is negotiated by the SMTP client and server. So the simplest way to activate this extension is to put the `trackingclient` and `trackingserver` on the defaults channel. However, if you wish to avoid use of this extension with systems outside the deployment, a more targeted approach is needed: Place `trackingclient` on every `tcp_*` channel that has `trackinginternal` set. Then place `trackingserver` on every channel that accepts messages from other hosts within the deployment. (Note that once again this may require configuration changes to separate internal and external traffic.)

48.1.3 MTQP server setup

An extended version of the MTRP protocol specified in [RFC 3887](#) is used to perform tracking and recall operations. If only tracking is desired a single MTQP server can be used for the entire deployment and can be run on any host. However, in order to recall messages an MTQP server must be running on every host with an MTA, and if "total recall" is enabled MTQP servers are required on all store hosts as well.

The MTQP server leverages existing MTA facilities which require a channel. Note that the MTQP channel doesn't sent or receive messages and has no rewrite rules or associated queue. The specification of such a channel is very simple:

```
mtqp smtp
mtqp-daemon
```

Although somewhat counterintuitive, the "[smtp](#)" channel option is required to indicate this is a channel with an associated server. Various optional channel options can also be specified, including:

- `maytllserver` or `musttllserver` are used to control whether SSL/TLS is allowed or required, respectively, before tracking/recall operations may be performed.
- `slave_debug` enables MTQP server debugging.
- `maysaslserver` may be specified to enable user authentication. User authentication is required in order to use the MTRACK and/or MRECALL commands.
- `disconnectbadauthlimit`, `disconnectbadcommandlimit`, and `disconnectcommandlimit` have their usual meanings.

Some MTQP-server-specific options are also available:

TOTAL_RECALL (boolean 0 or 1, default 0)	If set, this option enables recall of messages from the store. Note that if this is used it will result in messages being deleted from recipient's folder irrespective of whether or not they have been read. Use of this option should only be undertaken with a full understanding of the possible consequences.
AUTH_DEBUG (debug token list, default "")	This option is used to specify authentication debugging tokens.
COMMAND_TIMEOUT (integer seconds, default 60)	This option specifies the time the server will wait for a command before disconnecting.
STATUS_TIMEOUT (integer seconds, default 60)	This option specifies the time the server will wait for a status write to the client before disconnecting.
CUSTOM_BANNER_STRING (string, default "")	This option specifies the name the server uses to announce itself.

Additionally, the server recognizes and will honor the [TCP/IP channel-specific options](#) LOG_CONNECTION, TRACE_LEVEL, MAX_THREADS, and IGNORE_BAD_CERT.

Finally, a [Dispatcher](#) service definition for the MTQP server is also required:

```
[SERVICE=MTQP]  
PORT=1038  
IMAGE=IMTA_BIN:mtqp  
LOGFILE=IMTA_LOG:mtqp_server.log  
STACKSIZE=2048000
```

Chapter 49 TCP/IP channels

49.1 Typical TCP/IP channels and servers	49-4
49.2 SMTP SUBMIT servers	49-6
49.2.1 Enabling BURL support for SMTP SUBMIT	49-7
49.2.2 SMTP SUBMIT FUTURERELEASE support	49-12
49.3 LMTP channels	49-13
49.3.1 LMTP client TCP/IP channels	49-13
49.3.2 LMTP back end TCP/IP channel	49-14
49.4 TCP/IP-channel-specific options	49-16
49.4.1 552_PERMANENT_ERROR_STRING TCP/IP-channel-specific option	49-17
49.4.2 ALLOW_ETRNS_PER_SESSION TCP/IP-channel-specific option	49-18
49.4.3 ALLOW_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option	49-18
49.4.4 ALLOW_REJECTIONS_BEFORE_DEFERRAL TCP/IP-channel-specific option	49-19
49.4.5 ALLOW_SESSION_BLOCKS TCP/IP-channel-specific option	49-19
49.4.6 ALLOW_TRANSACTION_BLOCKS TCP/IP-channel-specific option	49-19
49.4.7 ALLOW_TRANSACTIONS_PER_SESSION TCP/IP-channel-specific option	49-20
49.4.8 ATTEMPT_TRANSACTIONS_PER_SESSION TCP/IP-channel-specific option	49-20
49.4.9 AUTH_DEBUG TCP/IP-channel-specific option	49-20
49.4.10 AUTH_PASSWORD, AUTH_USERNAME, EXTERNAL_IDENTITY TCP/IP- channel-specific options	49-20
49.4.11 BANNER_ADDITION TCP/IP-channel-specific option	49-21
49.4.12 BANNER_HOST TCP/IP-channel-specific option	49-21
49.4.13 BANNER_REVERSE_HOST TCP/IP-channel-specific option	49-21
49.4.14 BANNER_PURGE_DELAY TCP/IP-channel-specific option	49-21
49.4.15 BUFFER_SIZE TCP/IP-channel-specific option	49-22
49.4.16 CHECK_SOURCE TCP/IP-channel-specific option	49-22
49.4.17 CLIENT_CERT_NICKNAME TCP/IP-channel-specific option	49-22
49.4.18 COMMAND_RECEIVE_TIME TCP/IP-channel-specific option	49-22
49.4.19 COMMAND_TRANSMIT_TIME TCP/IP-channel-specific option	49-23
49.4.20 CONTINUATION_CHARS TCP/IP-channel-specific option	49-23
49.4.21 CUSTOM_VERSION_STRING TCP/IP-channel-specific option	49-23
49.4.22 DATA_RECEIVE_TIME TCP/IP-channel-specific option	49-23
49.4.23 DATA_TRANSMIT_TIME TCP/IP-channel-specific option	49-23
49.4.24 DISABLE_ADDRESS TCP/IP-channel-specific option	49-23
49.4.25 DISABLE_CIRCUIT TCP/IP-channel-specific option	49-24
49.4.26 DISABLE_EXPAND TCP/IP-channel-specific option	49-24
49.4.27 DISABLE_GENERAL TCP/IP-channel-specific option	49-25
49.4.28 DISABLE_SEND TCP/IP-channel-specific option	49-25
49.4.29 DISABLE_STATUS TCP/IP-channel-specific option	49-25
49.4.30 DOT_TRANSMIT_TIME TCP/IP-channel-specific option	49-26
49.4.31 FAST_SMTP_SESSION_TIME_LIMIT TCP/IP-channel-specific option ...	49-26
49.4.32 HELLO_RECEIVE_TIME TCP/IP-channel-specific option	49-26
49.4.33 HIDE_VERIFY TCP/IP-channel-specific option	49-26
49.4.34 IGNORE_BAD_CERT TCP/IP-channel-specific option	49-27
49.4.35 INITIAL_COMMAND TCP/IP-channel-specific option	49-27
49.4.36 LOG_BANNER TCP/IP-channel-specific option	49-27
49.4.37 LOG_CONNECTION TCP/IP-channel-specific option	49-27

49.4.38	LOG_TRANSPORTINFO TCP/IP-channel-specific option	49-28
49.4.39	MAIL_TRANSMIT_TIME TCP/IP-channel-specific option	49-28
49.4.40	MAILBOX_BUSY_FAST_RETRY TCP/IP-channel-specific option	49-29
49.4.41	MAX_A_RECORDS TCP/IP-channel-specific option	49-29
49.4.42	MAX_B_ENTRIES TCP/IP-channel-specific option	49-29
49.4.43	MAX_CLIENT_THREADS TCP/IP-channel-specific option	49-29
49.4.44	MAX_HELO_DOMAIN_LENGTH TCP/IP-channel-specific option	49-30
49.4.45	MAX_J_ENTRIES TCP/IP-channel-specific option	49-30
49.4.46	MAX_MX_RECORDS TCP/IP-channel-specific option	49-30
49.4.47	MAX_SERVER_THREADS TCP/IP-channel-specific option	49-31
49.4.48	OPEN_CONNECTION_TIME TCP/IP-channel-specific option	49-31
49.4.49	PACKET_SIZE_LIMIT TCP/IP-channel-specific option	49-31
49.4.50	PROXY_PASSWORD TCP/IP-channel-specific option	49-31
49.4.51	RCPT_TRANSMIT_TIME TCP/IP-channel-specific option	49-32
49.4.52	REJECT_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option	49-32
49.4.53	REUSE_TIMED_OUT_TRANSFERS TCP/IP-channel-specific option	49-32
49.4.54	SESSION_TIME TCP/IP-channel-specific option	49-33
49.4.55	Delay threshold TCP/IP-channel-specific option	49-34
49.4.56	SSL_CLIENT TCP/IP-channel-specific option	49-34
49.4.57	STARTTLS_FAILURE_RECONNECT_DELAY TCP/IP-channel-specific option	49-35
49.4.58	STATUS_DATA_RECEIVE_TIME TCP/IP-channel-specific option	49-35
49.4.59	STATUS_DATA_RECV_PER_ADDR_TIME TCP/IP-channel-specific option	49-35
49.4.60	STATUS_DATA_RECV_PER_BLOCK_TIME TCP/IP-channel-specific option	49-35
49.4.61	STATUS_DATA_RECV_PER_ADDR_PER_BLK_TIME TCP/IP-channel- specific option	49-35
49.4.62	STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option	49-36
49.4.63	STATUS_RCPT_RECEIVE_TIME TCP/IP-channel-specific option	49-36
49.4.64	STATUS_RECEIVE_TIME TCP/IP-channel-specific option	49-36
49.4.65	STATUS_TRANSMIT_TIME TCP/IP-channel-specific option	49-36
49.4.66	TRACE_LEVEL TCP/IP-channel-specific option	49-36
49.4.67	TRANSACTION_LIMIT_RCPT_TO TCP/IP-channel-specific option	49-36
49.4.68	TRANSACTION_TIME TCP/IP-channel-specific option	49-37
49.4.69	WINDDOWN_TIMEOUT TCP/IP-channel-specific option	49-37
49.5	AUTH_ACCESS mapping table	49-37
49.5.1	Local user entry in LDAP example	49-40
49.5.2	Remote user entry in LDAP example	49-40
49.5.3	Third party submission example	49-41
49.6	IP_ACCESS mapping table	49-43
49.7	TLS_ACCESS mapping table	49-45
49.8	Routing via gateway systems	49-46
49.8.1	Routing non-local mail to a mailhub	49-47
49.9	Blocking SMTP relaying	49-48
49.9.1	SRS and Relay Blocking	49-50
49.10	Triggering message transfer with remote SMTP systems	49-51
49.11	Authentication errors and resultant SMTP errors	49-52

TCP/IP channels are used to link the MTA to TCP/IP based networks such as the Internet, as well as linking multiple hosts within a single site. The TCP/IP channels all use either the Simple Mail Transfer Protocol (SMTP), or the similar Local Mail Transport Protocol (LMTP).

SMTP was defined originally in [RFC 821](#) and updated in [RFC 5321](#), and nowadays has many extensions such as those described in [RFC 1426](#), [RFC 1869](#), [RFC 870](#), and [RFC 1891](#). LMTP was defined in [RFC 2033](#).¹ The descriptions of these protocols and extensions may be found at the Internet Engineering Task Force (IETF) web site, www.ietf.org.

The TCP/IP SMTP channel includes a multithreaded SMTP server which runs under the control of the MTA [Dispatcher](#). Outgoing SMTP mail is processed by the multithreaded SMTP client channel program (`tcp_smtp_client`), run as needed under the control of the MTA [Job Controller](#).

The TCP/IP LMTP channel similarly includes a multithreaded LMTP server which runs under the control of the MTA [Dispatcher](#); indeed, on a back end LMTP Message Store host, usually the *only* components of the MTA that are used are the MTA Dispatcher and the MTA's LMTP server. The TCP/IP LMTP channel also includes a multithreaded LMTP client channel program (which is in fact the same code and program as the SMTP client channel program, merely configured a bit differently), run as needed (on full-blown MTA hosts front-ending an LMTP back end host) under the control of the MTA [Job Controller](#).

At most sites, TCP/IP channels are the primary channels in use. See [Typical TCP/IP channels and servers](#) for an overview of the basic, minimally-configured, TCP/IP channels and servers included by default in a Unified Configuration. There are many, many configuration modifications that may be made to TCP/IP channels. Besides the normal [Channel options](#) and especially the [SMTP and LMTP protocol channel options](#), see also the [TCPIP-channel-specific options](#). In legacy configuration, channel options were termed "channel keywords" and were set on channels in the `imta.cnf` file, whereas TCPIP-channel-specific options were set in channel-specific files such as `tcp_local_option`. In Unified Configuration, these fundamentally different categories of options may be distinguished by the location at which they are set, with channel options appearing directly under the name of the channel for which they are set,

```
msconfig> show role.channel:tcp_local.pool
role.channel:tcp_local.pool = SMTP_POOL
```

whereas TCP/IP-channel-specific options are grouped under `options`,

```
msconfig> show role.channel:tcp_local.options.*
```

For normal hosts on the Internet, configuring to prevent being an open SMTP relay, that is, configuring to [block open SMTP relaying](#), is an essential step. Nowadays initial installation typically takes care of the basics of such configuration, but reviewing that configuration is recommended.

Certain special configurations of TCP/IP channels can be of particular use. [Routing via gateway systems](#), describes how to establish a "gateway" channel to route mail through another system, as for instance to a mailhub or firewall system. [Triggering message transfer with remote SMTP systems](#) describes performing "polling" with TCP/IP channels, that is, requesting that a remote system attempt to deliver any stored messages to your system; for instance, this may be particularly useful over "intermittent" sorts of TCP/IP links, such as dial-up connections.

Note¹ When using LMTP to deliver messages to the Messaging Server Message Store, the MTA's LMTP server and LMTP client both make important use of several MTA proprietary LMTP extensions. These LMTP extensions are aimed to permit the LMTP server side of the LMTP transactions to be very simple, and not require "heavy weight" processing; the "heavier

weight" processing of message addresses and message content is instead pre-performed on "front" MTAs. Thus the MTA's LMTP server is designed and intended to work *only* when "front-ended" by an MTA LMTP client host. The MTA's LMTP client, in contrast, may in principle be used with any LMTP back end.

49.1 Typical TCP/IP channels and servers

In Unified Configuration, several basic `tcp_*` channels are established by default in the configuration. (Any legacy configuration generated in any relatively modern MTA version will also normally have several basic `tcp_*` channels defined.) *E.g.*:

```
msconfig> show role.channel:tcp_*
role.channel:tcp_local.official_host_name = tcp-daemon
role.channel:tcp_local.daemon = mailgate.domain.com
role.channel:tcp_local.identnonenumeric (novalue)
role.channel:tcp_local.inner (novalue)
role.channel:tcp_local.loopcheck (novalue)
role.channel:tcp_local.maysaslserver (novalue)
role.channel:tcp_local.maytlserver (novalue)
role.channel:tcp_local.nomx (novalue)
role.channel:tcp_local.pool = SMTP_POOL
role.channel:tcp_local.remotehost (novalue)
role.channel:tcp_local.saslswitchchannel = tcp_auth
role.channel:tcp_local.smtp (novalue)
role.channel:tcp_local.sourcetransitplugin = 1
role.channel:tcp_local.switchchannel (novalue)
role.channel:tcp_intranet.official_host_name = tcp_intranet-daemon
role.channel:tcp_intranet.allowswitchchannel (novalue)
role.channel:tcp_intranet.dequeueremoveroute (novalue)
role.channel:tcp_intranet.loopcheck (novalue)
role.channel:tcp_intranet.maysaslserver (novalue)
role.channel:tcp_intranet.maytlserver (novalue)
role.channel:tcp_intranet.mx (novalue)
role.channel:tcp_intranet.pool = SMTP_POOL
role.channel:tcp_intranet.saslswitchchannel = tcp_auth
role.channel:tcp_intranet.single_sys (novalue)
role.channel:tcp_intranet.smtp (novalue)
role.channel:tcp_intranet.sourcetransitplugin = 1
role.channel:tcp_submit.official_host_name = tcp_submit-daemon
role.channel:tcp_submit.maytlserver (novalue)
role.channel:tcp_submit.mustsaslserver (novalue)
role.channel:tcp_submit.saslswitchchannel = tcp_submit
role.channel:tcp_submit.smtp (novalue)
role.channel:tcp_submit.sourcetransitplugin = 1
role.channel:tcp_submit.submit (novalue)
role.channel:tcp_auth.official_host_name = tcp_auth-daemon
role.channel:tcp_auth.mustsaslserver (novalue)
role.channel:tcp_auth.smtp (novalue)
role.channel:tcp_auth.sourcetransitplugin = 1
role.channel:tcp_tas.official_host_name = tcp_tas-daemon
role.channel:tcp_tas.allowswitchchannel (novalue)
role.channel:tcp_tas.deliveryflags = 2
```

```

role.channel:tcp_tas.maytlserver (novalue)
role.channel:tcp_tas.mustsasserver (novalue)
role.channel:tcp_tas.smtp (novalue)
role.channel:tcp_tas.sourcetransitplugin = 1
msconfig>

```

with [Dispatcher options](#) (to complete the definitions of the SMTP and [SMTP SUBMIT](#) servers) of:

```

msconfig> show dispatcher.service:*MTP*
role.dispatcher.service:SMTP.image = IMTA_BIN:tcp_smtp_server
role.dispatcher.service:SMTP.logfilename = IMTA_LOG:tcp_smtp_server.log
role.dispatcher.service:SMTP.stacksize = 2048000
role.dispatcher.service:SMTP.tcp_ports = 25
role.dispatcher.service:SMTP_SUBMIT.image = IMTA_BIN:tcp_smtp_server
role.dispatcher.service:SMTP_SUBMIT.logfilename = IMTA_LOG:tcp_submit_server.log
role.dispatcher.service:SMTP_SUBMIT.parameter = CHANNEL=tcp_submit
role.dispatcher.service:SMTP_SUBMIT.stacksize = 2048000
role.dispatcher.service:SMTP_SUBMIT.tcp_ports = 587

```

This corresponds to what in legacy configuration would appear as channels in the `imta.cnf` file:

```

tcp_local daemon mailgate.domain.com identnonnumeric inner loopcheck \
  maysaslserver maytlserver nomx pool SMTP_POOL remotehost \
  saslsitchchannel tcp_auth smtp sourcespamfilter1 switchchannel
tcp-daemon

```

```

tcp_intranet allowswitchchannel dequeueremoveroute loopcheck maysaslserver \
  maytlserver ms pool SMTP_POOL saslsitchchannel tcp_auth single_sys smtp \
  sourcespamfilter1
tcp_intranet-daemon

```

```

tcp_submit maytlserver mustsasserver saslsitchchannel tcp_submit smtp \
  sourcespamfilter1
tcp_submit-daemon

```

```

tcp_auth mustsasserver smtp sourcespamfilter1
tcp_auth-daemon

```

```

tcp_tas allowswitchchannel deliveryfalgs 2 maytlserver mustsasserver smtp \
  sourcespamfilter1
tcp_tas-daemon

```

And the corresponding legacy configuration Dispatcher part of the configuration -- the SMTP and [SMTP SUBMIT](#) servers -- would be:

```

!
! multithreaded SMTP server
!
[SERVICE=SMTP]
PORT=25
IMAGE=IMTA_BIN:tcp_smtp_server

```

```

LOGFILE=IMTA_LOG:tcp_smtp_server.log
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 2476 Submit server
!
[SERVICE=SMTP_SUBMIT]
PORT=587
! Uncomment the following line if you want to support SSL on the alternate port 465
!TLS_PORT=465
IMAGE=IMTA_BIN:tcp_smtp_server
LOGFILE=IMTA_LOG:tcp_submit_server.log
PARAMETER=CHANNEL=tcp_submit
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=

```

At sites using LMTP, so-called "front end" MTAs typically have, in addition to the TCP/IP channels shown above, one or more [LMTP client channels](#), typically with `tcp_lmtpcs*` sorts of names, appearing in Unified Configuration as:

```

msconfig> show channel:tcp_lmtpcs*
role.channel:tcp_lmtpcs.official_host_name = lmtpcs-daemon
role.channel:tcp_lmtpcs.connectcanonical (novalue)
role.channel:tcp_lmtpcs.defragment (novalue)
role.channel:tcp_lmtpcs.dequeueremoveveroute (novalue)
role.channel:tcp_lmtpcs.fileinto = @$40:$U+$S@$D
role.channel:tcp_lmtpcs.lmtp (novalue)
role.channel:tcp_lmtpcs.multigate (novalue)
role.channel:tcp_lmtpcs.nomx (novalue)
role.channel:tcp_lmtpcs.pool = SMTP_POOL
role.channel:tcp_lmtpcs.port = 225
role.channel:tcp_lmtpcs.single_sys (novalue)

```

Or in legacy configuration in the `imta.cnf` file appearing as:

```

!
! tcp_lmtpcs (LMTP client - store)
tcp_lmtpcs defragment lmtp multigate connectcanonical fileinto @$40:$U+$S@$D \
  port 225 nomx single_sys pool SMTP_POOL dequeue_removeveroute
lmtpcs-daemon

```

So-called "back end" LMTP systems run a simplified MTA configuration having only one (LMTP server) channel as discussed in [LMTP back end TCP/IP channel](#).

49.2 SMTP SUBMIT servers

A TCP/IP channel marked with the `submit` channel option will function with respect to message submission as an SMTP SUBMIT server, as defined in [RFC 6409 \(Message](#)

[Submission for Mail](#)). Port 587 is reserved for SMTP SUBMIT use, so `submit` is [normally set on the channel](#) that, in the [Dispatcher](#) configuration, is associated with port 587; see the `dispatcher.tcp_ports` option (Unified Configuration). But SMTP SUBMIT service can also be set up on a different or other ports, if desired: any incoming channel marked with `submit` will operate as an SMTP SUBMIT server. [RFC 5068 \(Email Submission Operations: Access and Accountability Requirements\)](#), also known as BCP 134, encourages sites to support—and indeed transition to using—SMTP SUBMIT rather than regular SMTP for initial submission of user messages.

The configuration established when the MTA is installed normally includes an SMTP SUBMIT server in the form of a `tcp_submit` channel; this channel is marked with the `submit` channel option, and the Dispatcher configuration sets it to listen on port 587. In accord with [RFC 5068](#)'s recommendations, the channel is also marked with `mustsaslsrver` and `maytlssrver`, meaning that users *must* authenticate to submit messages to this channel and *may* use TLS. See [Typical TCP/IP channels and servers](#) for an example. (TLS use should be encouraged, but is not outright required in this typical configuration; sites that prefer to also require TLS use may instead set `musttlssrver`.)

Sites should encourage their local users to submit messages to the SMTP SUBMIT server on port 587, rather than to the regular SMTP server on port 25. But note that this does also imply that users *must* authenticate and *ought* to use TLS—some configuration of the users' clients may be needed to achieve this.

Extensions to SMTP SUBMIT permit additional functionality for users and user clients: see the [BURL_ACCESS mapping table](#) and [SMTP SUBMIT FUTURERELEASE support](#).

49.2.1 BURL_ACCESS mapping table

As of JES Messaging Server 7.0-0.04, the MTA supports the BURL extension to SMTP SUBMIT, defined in [RFC 4468 \(Message Submission BURL Extension\)](#), and the Message Store IMAP server supports [RFC 4467 \(IMAP - URLAUTH Extension\)](#). BURL permits an email client to refer, in submitted messages, to content to be retrieved (using the IMAP URLAUTH extension) from an IMAP server. This allows email clients to submit messages including content without having to first download that content to the client and then upload (submit it) directly to the SMTP SUBMIT server itself: to include content by supplying a URL reference (including an authentication token allowing access) to the location of the content on an IMAP server. Availability of the BURL extension is controlled by the `BURL_ACCESS` mapping table, discussed below, and the [MTA options `imap_username` and `imap_password`](#).

For the BURL extension to be made available, a `BURL_ACCESS` mapping table must be defined. Note that BURL is specifically an [SMTP SUBMIT](#) feature; in terms of MTA configuration, only channel(s) marked with the [submit channel option](#) are capable of offering BURL support.

Technical note: A client BURL command to an SMTP SUBMIT server for which BURL has not been enabled will be rejected with the error:

```
503 5.5.1 BURL has not been enabled.
```

BURL is only supported (may only be enabled) on SMTP SUBMIT channels; a client BURL command to an LMTP server will be rejected with the error:

```
503 5.5.0 BURL illegal on LMTP port.
```

while a client BURL command to an SMTP (non-SUBMIT) server will be rejected with the error:

```
503 5.5.1 BURL only allowed on submission port.
```

The BURL_ACCESS mapping table controls, via two different probe strings, first whether the BURL extension is advertised, and then second, whether a client's BURL command is accepted. The first probe is performed before responding to an EHLO command. It has the form:

```
port_access-probe-info|channel|authentication-identity|
```

Here `port_access-probe-info` consists of all the information usually included in a [PORT_ACCESS mapping table](#) probe. It will be blank if BURL is being used in a "disconnected" context such as batch SMTP. `channel` is the current source channel. `authentication-identity` is the user's canonical authenticated login identity, normally `uid@canonical-domain` (that is, the user's LDAP `uid` attribute value, an at sign, and the canonical domain name in which the user's entry resides as found during domain lookup). `authentication-identity` will be blank if no authentication has been performed. The "A" input flag will be set if SASL authentication has been performed, and the "T" input flag will be set if TLS is in use; thus the mapping templates may test using `$.A` and `$.T`, respectively. In order to offer BURL support, the mapping output for this first probe must set `$.Y`; it may also optionally provide a space-separated list of supported URL types. `imap` is assumed if no explicit string is returned.

The second probe is performed when a BURL command is actually sent by the SUBMIT client and received by the SMTP SUBMIT server. In addition to the prior fields (described above), this second probe also includes as a final `client-url` field the URL specified in the client's BURL command:

```
port_access-probe-info|channel|authentication-identity|client-url
```

where `client-url` would normally be in URLAUTH syntax as defined in [RFC 4467 \(IMAP - URLAUTH Extension\)](#). See [RFC 4467](#) for details on URLAUTH syntax, but for a rough idea to better understand the remainder of this discussion, consider that for BURL "submit user" access, the `client-url` will usually take a form along the lines of:

```
imap://enc-user@hostport/optional-expire-clause;URLAUTH=submit+enc-user:mechanism:token
```

(Other types of access in a URLAUTH and hence other forms of `client-url`, such as for "anonymous" access, are possible, but their use is more questionable--see the "Security Considerations" section of [RFC 4468](#) -- and therefore the remainder of this discussion will focus on enabling only "submit user" access.)

In this second (actual BURL command) BURL_ACCESS probe, as in the prior (advertise BURL) probe, the "A" input flag will be set if SASL authentication has been performed, and the "T" input flag will be set if TLS is in use; thus the mapping templates may test using `$.A` and `$.T`, respectively.

Additionally, for this second (actual received BURL command) probe, the vertical bar flag, `|`, will be set if the original URL sent by the client contained any vertical bars (which if present could possibly confuse some sorts of access checks), and thus the mapping entry template may test for vertical bars in the original client URL using the `$.|` test; note that the [mapping_paranoia MTA option](#), if set, will cause any vertical bar within the client's original URL to be replaced in the probe by the specified alternate character (with the vertical bar input flag still being set). The mapping must set `$.Y` for the URL to be accepted for processing. If `$.D`

is also set, then the string result of the mapping replaces the client's entire originally specified URL. New in 7.4-18.01, if \$M is set, then the IMAP host name in the client's original URL will be replaced by the mailHost of the currently authenticated user; this tends to provide both better security (by enforcing that BURL connections will only be made to a site's own mailHost host(s)), and better performance (by more likely connecting to the "correct" host), than relying on the IMAP host name supplied by the user's client. See [Table of BURL_ACCESS mapping flags](#) for a summary of these available flags and tests.

Table 49.1 BURL_ACCESS mapping flags

Flag	Description
\$Yurl-types	For the initial (EHLO) probe, enable advertising BURL support, optionally providing via the string argument url-types a space-separated list of the URL types to advertise.
\$Nstring	For the later (BURL command) probes, reject access. If the optional string is supplied, use it as the (entire) SMTP rejection, including SMTP error code and extended code as well as text. If no such string is supplied, then the default SMTP error used for such rejections is 533 5.7.1 Access denied to specified URL.®
\$I	(New in 7.4-18.01) For the later (BURL command) probes, if specified in an entry with \$N rejecting that BURL command, the \$I means further to forcibly disconnect the session with disconnect reason text "BURL_ACCESS forced disconnect".®
\$Y	For the later (BURL command) probes, a plain \$Y flag allows the BURL command.
\$Dnew-url	For the later (BURL command) probes, if \$Y was also specified (allowing access), then use the specified new-url instead of the URL that the SMTP SUBMIT client provided.®
\$M	(New in 7.4-18.01) For the later (BURL command) probes that succeed, in the actual BURL command override the IMAP host name in the client-provided URL and instead connect to the host of the mailHost attribute of the currently authenticated user. The BURL command probe itself will be considered to fail if the currently authenticated user has no mailHost attribute set.®
\$T	(New in Messaging Server 7.0.5) For the later (BURL command) probes that succeed, force TLS use for opening the BURL URL (force TLS use in the connection to fetch the part specified in the client's BURL URL).®
\$X	(New in Messaging Server 7.0.5) For the later (BURL command) probes that succeed, forcibly disable TLS use for opening the BURL URL (force non-use of TLS in the connection to fetch the part specified in the client's BURL URL).®
Flag comparisons	Description
\$:	Match only if the original client URL included the vertical bar character, ®
\$;	Match only if the original client URL included no vertical bar character, ®
\$:A	Match only if SASL (SMTP AUTH) was used.
;\$A	Match only if SASL (SMTP AUTH) was not used.
\$:T	Match only if TLS was used.

Enabling BURL support for SMTP SUBMIT

\$;T	Match only if TLS was not used.
------	---------------------------------

Ⓜ Available for the later (BURL command containing a URL) probes only; not available for the initial probe on whether or not to offer the BURL extension

At an absolute minimum, a site's BURL_ACCESS mapping table should be configured to verify that a proper type of URL has been specified: typically only `imap:` URLs should be allowed. Additionally, in the case of IMAP URLs used in SMTP SUBMIT message submission, a check ought to be made to insure that the URL "belongs" to the user: that is, that the access user in the URL matches the authenticated uid for the SUBMIT session. Indeed, typically sites will not even want to advertise BURL in the SMTP SUBMIT server response unless and until the client has authenticated--in terms of the BURL_ACCESS mapping table, this means to start with an entry that enables advertising BURL only if the authentication-identity field in the initial probe is non-empty. Additionally, it is almost always essential to restrict message fetching access via a BURL command's `imap:` URL to an appropriate set of IMAP servers. As of 7.4-18.01, use of the `$M` flag in the mapping template (right hand side) is a simple way to enforce that only users' own `mailHost` values are used as the IMAP server in the eventually executed BURL command. Prior to 7.4-18.01, the BURL_ACCESS mapping table should typically be setup up to explicitly look for (match) a list of known, internal IMAP servers (users' valid `mailHost` values) and only allow BURL commands using those IMAP server host names.

Thus as of 7.4-18.01, a minimal BURL_ACCESS mapping table might be something like:

BURL_ACCESS

```
! Initial entry to allow advertising BURL in EHLO response
*|tcp_*|%*|                               imap$Y
! Allow BURL commands, connecting to user's own mailHost
*|*.*|imap://*;URLAUTH=submit+$1*:*       $:A$M$Y
```

Or if hosted domains are in use, then include an additional entry to match the hosted domain user use:

BURL_ACCESS

```
! Initial entry to allow advertising BURL in EHLO response
*|tcp_*|%*|                               imap$Y
! Allow BURL commands, connecting to (default domain) user's own mailHost
*|*.*|imap://*;URLAUTH=submit+$1*:*       $:A$M$Y
! Allow BURL commands, connecting to (hosted domain) user's own mailHost
*|*.*|imap://*;URLAUTH=submit+$1*%40$2*:* $:A$M$Y
```

A better minimal BURL_ACCESS mapping table, implementing the recommended security provisions of [RFC 4468](#) (that is, also requiring TLS encryption as well as SMTP AUTH authentication in order to allow BURL), would be (as of the 7.4-18.01 version):

BURL_ACCESS

```
! Initial entry to allow advertising BURL in EHLO response, if TLS was used
*|tcp_*|%*|                               $:T$Yimap
```

```
! Allow BURL commands, connecting to (default domain) user's own mailHost
*|*.*|imap://*;URLAUTH=submit+$1*:* $:AS:T$M$Y
! Allow BURL commands, connecting to (hosted domain) user's own mailHost
*|*.*|imap://*;URLAUTH=submit+$1*%40$2*:* $:AS:T$M$Y
```

In versions prior to 7.4-18.01, a minimal BURL_ACCESS mapping table would be more verbose. For instance, the following makes use of a subsidiary X-IMAP-HOSTS mapping table to list a site's valid IMAP server host names.

X-IMAP-HOSTS

```
mail1.example.com $Y
mail2.example.com $Y
mail3.example.com $Y
```

BURL_ACCESS

```
! Initial entry to allow advertising BURL in EHLO response
*|tcp_*|*.*|imap$Y
! Allow BURL commands that request connecting to one of the "known" IMAP hosts
! listed in the X-IMAP-HOSTS mapping table
*|*.*|imap://*.*/*;URLAUTH=submit+$1*:* $C$:AS|X-IMAP-HOSTS;$4|$Y$E
! Ditto for users in hosted domains
*|*.*|imap://*.*/*;URLAUTH=submit+$1*%40$2*:* $C$:AS|X-IMAP-HOSTS;$4|$Y$E
```

Once the SMTP SUBMIT server has checked that BURL use should be allowed in a particular context, then in order to perform the IMAP URLAUTH operation specified in a BURL command, the SMTP SUBMIT server has to have the ability to log in to the IMAP server as the submit user. The [imap_username](#) and [imap_password](#) MTA options are used to accomplish this. [imap_username](#) specifies the submit user and defaults to the setting of the [submituser](#) IMAP option (in legacy configuration, the `service.imap.submituser` configutil option) if not specified. [imap_password](#) specifies the password which of course must match the value set for the submit user account. The [imap_password](#) option has no default value.

Technical note: Once logged in as the submit user, then the BURL check of the hashed authentication token in the URLAUTH is performed: only messages accessible to the user issuing the BURL command will be fetched. That is, the submit user logs in, but it is not the submit user's message access that determines whether a message or message part can be fetched and incorporated but rather the access of the original user is validated from the URLAUTH.

When using BURL to fetch an entire, pre-composed message, a BURL command replaces the usual DATA command in an SMTP dialogue. Alternatively, when the SMTP extension CHUNKING is supported (see [RFC 3030](#) and the [chunking* channel options](#)), then BURL and BDAT commands may be interlaced---meaning that a new message may be composed incorporating material (for instance, attachments) fetched via BURL commands. Unless explicitly disabled (see [nochunking* channel options](#)), the MTA's TCP/IP channels (and in particular its SMTP SUBMIT servers) by default support CHUNKING.

Note that new in 7.4-18.01, the MTA has a feature to force disconnection of the SMTP SUBMIT session if a user attempts "too many" bad (failed) BURL commands: see the [disconnectbadburllimit channel option](#).

Note that as regards MTA message transaction logging, a message submitted using BURL will include a "U" modifier on its "E" enqueue record, or include "UC" if both BURL and CHUNKING were used; see [MTA transaction log entry format](#).

49.2.2 SMTP SUBMIT FUTURERELEASE support

New in Messaging Server 7.0-0.04, the MTA supports the FUTURERELEASE extension to SMTP SUBMIT, defined in [RFC 4865 \(SMTP Submission Service Extension for Future Message Release\)](#). FUTURERELEASE permits a client to submit a message to the SMTP SUBMIT server for the server to hold onto in its (server) queue until releasing for delivery at a specified time in the future. This can be useful for clients that wish to compose messages "ahead of time", that will only become eligible for delivery at some later time, or which have issues with saving/retaining messages locally themselves, or with ensuring submission at the "right time". A typical use case might be for messages intended as announcements or event reminders.

FUTURERELEASE support is enabled by placing the [futurerelease channel option](#) on a [submit](#) source channel used for initial message submission. The [futurerelease channel option](#) takes a single integer argument: the maximum number of seconds a message can be held. Note that FUTURERELEASE is specifically an [SMTP SUBMIT](#) feature: in terms of MTA configuration, only channel(s) marked with the `submit` keyword are capable of offering FUTURERELEASE support. A client attempt to use a FUTURERELEASE parameter in a MAIL FROM command to an SMTP SUBMIT server for which FUTURERELEASE has not been enabled will be rejected with the error:

```
504 5.7.4 FUTURERELEASE extension not available.
```

or

```
503 5.7.1 FUTURERELEASE extension not available.
```

Care should be used when enabling future release since it allows messages to be in effect stored in the MTA's queues. FUTURERELEASE should only be used for channels handling initial message submission and authentication of the message submitter should be required.

Additionally, if you wish to make distinctions about just which senders may use FUTURERELEASE (or for how long different classes of senders may postpone message delivery), consider also using the `mailSMTPSubmitChannel` user attribute (or as of 8.0, whatever LDAP attribute is named by the `ldap_auth_attr_submit_channel` MTA option) to sort different classes of senders into different source channels with appropriately different [futurerelease](#) settings.

Note that similar functionality was previously available: specification of a Deferred-delivery: header field in a submitted message coupled with use of the `deferred` channel keyword on the destination channel provided the ability to defer delivery of messages. However, FUTURERELEASE provides superior functionality:

1. The FUTURERELEASE facility is controlled by a setting on the source channel, allowing it to be provided to a subset of the user population. In contrast, placing `deferred` on a destination channel opened the door to *anyone* submitting a message destined for that channel to have that message be held for some period of time.
2. There's no way for a client which sets a Deferred-delivery: header field to know whether or not the header has actually caused the message to be deferred. The FUTURERELEASE SMTP SUBMIT extension, on the other hand, lets the client know how long a message can

be held and an error will be returned to the client if the message cannot be held for the time the client requested.

3. With Deferred-delivery:, there was no (automatic) way to place a limit on the amount of time a message could be held. Instead what (normally) happened was that a message held longer than the channel's last `notices` value would simply be returned as undeliverable.
4. Deferred-delivery: settings on messages did not survive a Job Controller restart.

As part of the implementation work for `FUTURERELEASE`, the old Deferred-delivery: mechanism has been redesigned to address some (but not all) of these points. In particular, the deferred channel keyword has been replaced by two new channel options, `deferredsource` and `deferreddestination`. (The `deferred` option is now a synonym for `deferreddestination`.) Both of these options accept an optional integer argument specifying in seconds the maximum amount of time in the future a Deferred-delivery: header can specify and still be honored. The default if no argument is specified is 60*60*24*7, or 7 days. `deferredsource` enables Deferred-delivery: processing on the basis of the source channel while `deferreddestination` operates on destination channels. Finally, Deferred-delivery: settings on messages now survive Job Controller restarts. This addresses all of the points on the above list except (2): use of a Deferred-delivery: header field still provides no mechanism for informing the client whether or not their request will be honored.

Note that, as discussed above, `FUTURERELEASE` is an extension defined (and supported by the MTA) solely for use with `SMTP SUBMIT`. A client attempt to use `FUTURERELEASE` to the MTA's SMTP server will result in the error:

```
504 5.7.4 FUTURERELEASE is a SUBMIT extension; it cannot be used in SMTP.
or
```

```
503 5.7.1 FUTURERELEASE extension not available.
```

49.3 LMTP channels

LMTP over TCP/IP channels are a special case of [TCP/IP channels](#). The MTA's LMTP channels come in two varieties, [LMTP client](#) and [LMTP server](#). (Unlike the the MTA's general SMTP over TCP/IP channels which are often configured for bidirectional use, the MTA's LMTP channels are each dedicated to either client *or* server operation, the LMTP client channels operating on "front end" MTAs to transmit via LMTP over TCP/IP to back end Message Store systems, and the LMTP server operating on back end Message Store systems to deposit messages into the Message Store.)

49.3.1 LMTP client TCP/IP channels

In the MTA's implementation of LMTP, an LMTP client channel is a specially configured [TCP/IP channel](#). At its most basic, any outbound TCP/IP channel configured with the `lmtcp` channel option is an LMTP client channel. However, additional channel options should also be set upon an LMTP client channel; see the example in [Typical TCP/IP channels and servers](#).

In the Messaging Server implementation, the intended design is that, to the greatest extent possible, message processing (including [address canonicalization](#), [spam/virus filter package processing](#), [user Sieve filter application](#), [defragmentation of MIME message fragments](#), any content conversion implemented via the [conversion channel](#), *etc.*), will be applied on the "front

end" MTAs during (or prior to) enqueue to a `tcp_lmtpcs*` channel. (Then the [LMTP server](#) on the back end Message Store host need merely insert the already-entirely-pre-digested message into the Message Store.) In particular:

- In a typical LMTP configuration, various address transformations resulting from the interpretation of the [ldap_delivery_option](#) MTA option and coordinating LMTP-oriented rewrite rules are usually best handled by setting the [multigate](#) and [connectcanonical](#) channel options on `tcp_lmtpcs*` channels.
- Various TCP/IP or LMTP protocol relevant channel options, *e.g.*, [port](#) and [single_sys](#), are commonly set.
- To ensure MIME defragmentation of any messages destined via LMTP to the Message Store, the [defragment](#) channel option should be set on the `tcp_lmtpcs` channels.
- And so that application of Sieve filters on the "front end" MTAs will properly convey any intended "fileinto" action effects back to the back end LMTP server, the [fileinto](#) channel option should be set on the `tcp_lmtpcs` channel(s).
- (The MTA's LMTP implementation also makes use of proprietary LMTP extensions to convey other important information from the LMTP client to the LMTP server, including quota data and IMAP flag Sieve effects; the MTA's LMTP server and LMTP client negotiate the extension use and send this information automatically.)

In the MTA implementation, the LMTP client channel code *is* the SMTP client channel code, merely configured specially: the many [channel options](#) and [TCP/IP-channel-specific options](#) affecting SMTP client operation may also be set and affect LMTP client operation. (Note that this in contrast to the MTA's [LMTP server](#), which consists of distinct and separate code from the SMTP server, so configuration options for the SMTP server do not always have LMTP server counterparts.)

49.3.2 LMTP back end TCP/IP channel

On an LMTP back end Message Store host, the [typical TCP/IP channels and servers](#) are replaced instead by an LMTP server, defined in Unified Configuration as a `tcp_lmtpss` channel and corresponding LMPSS service under the Dispatcher:

```
msconfig> show channel:tcp_*
role.channel:tcp_lmtpss.official_host_name = tcp_lmtpss-daemon
role.channel:tcp_lmtpss.flagtransfer (novalue)
role.channel:tcp_lmtpss.identnonnumeric (novalue)
role.channel:tcp_lmtpss.lmtp (novalue)
msconfig> show dispatcher.service:*MTP*
role.dispatcher.service:LMPSS.image = IMTA_BIN:tcp_lmtp_server
role.dispatcher.service:LMPSS.logfilename = IMTA_LOG:tcp_lmtpss_server.log
role.dispatcher.service:LMPSS.parameter = CHANNEL=tcp_lmtpss
role.dispatcher.service:LMPSS.stacksize = 2048000
role.dispatcher.service:LMPSS.tcp_ports = 225
msconfig> show mapping:PORT_ACCESS
role.mapping:PORT_ACCESS.rule TCP|*|225|*|* $C$|INTERNAL_IP;$1|$Y$E
role.mapping:PORT_ACCESS.rule TCP|* $N500 Do not connect to this machine
msconfig> show mapping:INTERNAL_IP
role.mapping:INTERNAL_IP.rule internal-host-or-subnet $Y
role.mapping:INTERNAL_IP.rule another-internal-host-or-subnet $Y
```

```
...
role.mapping:INTERNAL_IP.rule ${:::1} $Y
role.mapping:INTERNAL_IP.rule 127.0.0.1 $Y
role.mapping:INTERNAL_IP.rule * $N
```

In legacy configuration, this would be defined as a `tcp_lmtpss` channel in the `imta.cnf` file:

```
tcp_lmtpss flagtransfer identnonnumeric lmtp
tcp_lmtpss-daemon
```

and a Dispatcher definition of the LMTTPSS server in the `dispatcher.cnf` file:

```
!
! rfc 2033 LMTTP server - store
!
[SERVICE=LMTTPSS]
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=
```

and with `PORT_ACCESS` and `INTERNAL_IP` mapping tables in the `mappings` file:

```
PORT_ACCESS

TCP|*|225|*|*    $C$|INTERNAL_IP;$1|$Y$E
TCP|*            $N500 Do not connect to this machine

INTERNAL_IP

    host's-own-public-IP                $Y
    another-public-IP-for-host          $Y
    ...
    internal-host-or-subnet             $Y
    another-internal-host-or-subnet     $Y
    ...
    ${:::1}                             $Y
    127.0.0.1                           $Y
    *                                    $N
```

49.3.2.1 LMTTP server options

The `loglevel` option may be set under the `tcp_lmtp_server` group to override for [LMTTP server purposes](#) any MTA-wide setting (`mta.loglevel`) of the `nslog` log level (controlling any Message Store logging triggered from the LMTTP server).

Note that all other options affecting [LMTP server](#) operation are instead either set as [channel options](#) (see especially the [SMTP and LMTP protocol channel options](#)), or as [TCP/IP-channel-specific options](#). In particular, the [logging](#) channel option on the LMTP server channel, normally `tcp_lmtpss`, controls the MTA side of LMTP server logging.

49.3.2.1.1 `loglevel` Option Under `mta`

The `loglevel` option for the MTA specifies an MTA log level used for the `imta` log file (primarily used by `ims_master` and the [LMTP server](#)). Its value can be one of `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information`, or `debug`.

Note that this MTA-wide setting of `mta.loglevel` (by default affecting both `ims-ms` channels and the LMTP server) can be overridden for the [LMTP server](#) via a `tcp_lmtp_server.loglevel` setting.

49.4 TCP/IP-channel-specific options

[TCP/IP channels](#) support, in addition to the usual [channel options](#), a number of TCP/IP-channel-specific options. These TCP/IP-channel-specific options are set in legacy configuration in a channel-specific option file, or in Unified Configuration are set under the channel's `options` option. For instance:

```
msconfig> set channel:tcp_local.options.BANNER_PURGE_DELAY 200
```

Note that in Unified Configuration such channel-specific options (those set under the `options` option) are not (currently) schema checked: be careful when setting them as `msconfig` will not warn of invalid values or syntax as it would for regular channel options! (Nor will `msconfig` show whether or not such channel-specific options have any default value.)

In legacy configuration, where an option file is used, such an option file must be named `x_option` where `x` is the name of the channel, and stored in the MTA table directory. Since the name of the default channel for the SMTP server is usually `tcp_local`, the option file for the SMTP server is usually `/opt/SUNWmsgsr/config/tcp_local_option` on UNIX; similarly, the name of the default channel for the SMTP SUBMIT server is usually `tcp_submit`, so the option file for the SMTP SUBMIT server is usually `/opt/SUNWmsgsr/config/tcp_submit_option`.

For incoming messages, such TCP/IP-channel-specific options are actually options for the server (SMTP, SMTP SUBMIT, or LMTPSS) as a whole. In particular, the only channel options that matter for incoming SMTP submissions are those for the SMTP server's default channel, not any channel options for possible other channels that may putatively handle the incoming message due to a `*switchchannel` channel option based "switching". Thus for instance in a typical configuration, for all incoming SMTP messages to port 25, it is only the channel-specific options set under `channel:tcp_local.options` that matter (in legacy configuration, only the `tcp_local_option` file that matters); similarly, the channel-specific options set under `channel:tcp_submit.options` affect all SMTP SUBMIT submissions (in legacy configuration, the `tcp_submit_option` file affects all SMTP SUBMIT submissions); and the channel-specific options set under `channel:tcp_lmtpss.options` affect all messages accepted by the LMTP server (in legacy configuration, the `tcp_lmtpss_option` file affects all messages accepted by the LMTP server). Any other channel-specific options (in legacy configuration, any other `tcp_*_option` file(s)) would only be potentially relevant for outgoing SMTP or LMTP messages---only affect SMTP or LMTP client operation.

Note that while master channel programs (the outgoing channel direction or SMTP client) read their option file each time they run and usually do not keep running for an especially long time, a slave channel program (each SMTP, SMTP SUBMIT, or LMTP server process) reads the option file only when it is first started, hence will not see changes while it continues to run, and its running time may be in the scale of hours. Server processes do automatically shut down periodically, and the Dispatcher creates new server processes in replacement, as required; so changes affecting server processes can get seen eventually, even in the absence of an explicit restart of the relevant server processes---but the change is likely not to take effect "quickly" in the absence of an explicit restart.

Note also that as of Messaging Server 7.0, TCP/IP channel option files for SMTP channels and LMTP client channels (but *not* yet LMTP server channels) are incorporated as part of a compiled configuration, if one exists. So as of Messaging Server 7.0, if a compiled configuration exists, it is necessary to recompile to get changes seen:

```
# imsimta cnbuild
```

Then, no different than ever, when it is important or essential to get changes to take effect *quickly*, as opposed to waiting for existing server processes and channel delivery processes to naturally age, executing

```
# imsimta restart *
```

will restart (the otherwise often quite long-running) SMTP, SMTP SUBMIT, and LMTP server processes; and see the `imsimta qm` utility's `stop` and `start` commands, or alternatively its `stress` and `unstress` commands, which may be used to force quick replacement of the (usually relatively short-lived in any case) SMTP and LMTP client processes (master channel jobs).

Note² Most of the options described actually relate to the SMTP protocol itself, rather than to the TCP/IP transport. As such, other MTA channels that use the SMTP protocol over other transports may have similar options.

49.4.1 TCP/IP-channel-specific options: 552_PERMANENT_ERROR_STRING (string)

[RFC 821](#) showed 552 as a temporary error at a RCPT TO: command (although in all other cases 500 numbers are permanent errors); some servers have implemented inconsistent use of the 552 error number. The MTA's SMTP client by default, if this option is not set, normally interprets 552 as a temporary error when seen as a response to an address command (MAIL FROM:, RCPT TO:, or VRFY:); but see below for a new-in-6.3 refinement of the MTA's behavior when a 5xy response is seen to the *first* RCPT TO: command in a transaction. If this SMTP client option is set, then when an 552 response is received to an address command (MAIL FROM:, RCPT TO:, or VRFY), then the string associated with it (excluding the initial "552 ", but including any leading *x.y.z* extended error code) is compared against this string. If, and only if, the string matches will the 552 be treated as permanent. Case is not significant in the comparison. For instance, if dealing with a remote server that will return "552 user no longer present" as an "intended" permanent error, then this option could be set to:

```
552_PERMANENT_ERROR_STRING=user no longer present
```

This option is not supported by the LMTP client; it is an SMTP client-only option.

New in 6.3, if the MTA's SMTP client sees any 5xy response including a 552 response to its *first* RCPT TO: command in a transaction and where after issuing its error the remote SMTP server then immediately disconnects the session, (which behavior is, by the way, a standards violation on the part of the remote SMTP server), then the MTA (regardless of the setting of this option) will unconditionally interpret this as a permanent error for that recipient, but will reenqueue any additional recipients of that message (copy) for another delivery attempt.

New in 7.0, the option value may use [glob-style wildcards](#); this new feature is available to aid in cases where a remote server's error response includes the input address, hence where a static, fixed value does not provide adequate matching.

49.4.2 TCP/IP-channel-specific options: ALLOW_ETRNS_PER_SESSION (integer)

This SMTP server option sets a limit on the number of ETRN commands accepted per session. The default is 1. When the limit is exceeded, the SMTP server will issue an error response to any additional ETRN command of (prior to MS 7.0.5):

```
550 5.7.1 ETRN session limit reached.
```

or as of MS 7.0.5:

```
458 4.7.1 ETRN session limit reached
```

See also the [*etrn channel options](#) for a discussion of channel options affecting the MTA's response to ETRN commands.

49.4.3 TCP/IP-channel-specific options: ALLOW_RECIPIENTS_PER_TRANSACTION (integer)

This SMTP/LMTP server option sets a limit on the number of recipients allowed per message, and on the number of address verifications (VRFY: commands) permitted during a transaction. (Note that the count of actual recipients, RCPT TO:, is separate from the count of verifies, VRFY:; that is, VRFY:s do not count against the RCPT TO: limit, nor do RCPT TO:s count against the VRFY: limit; each is limited *independently* to the ALLOW_RECIPIENTS_PER_TRANSACTION value.) Any additional recipients will be rejected with a temporary error response to the RCPT TO: command:

```
451 4.5.3 Too many recipients specified
```

Any additional address verification attempts will be rejected with a temporary error

```
451 4.5.2 Verification blocked; too many operations performed
```

at the VRFY: line. The default is 128. (In iMS 5.2 and earlier, the default was no limit.) See also the [recipientlimit](#) and (in the case of SMTP) [disconnectrecipientlimit channel options](#), and the [ldap_recipientlimit](#) and [ldap_domain_attr_recipientlimit](#) MTA options. Note that with any setting of ALLOW_RECIPIENTS_PER_TRANSACTION other

than the maximum allowed integer (2147483647), the SMTP server will not allow multiple, comma-separated values on the RCPT TO: command line. (The LMTP server, in contrast, will always allow multiple, comma-separated values on the RCPT TO: command line --- though the MTA's LMTP client will not itself send such a RCPT TO command.)

Note also that, unlike a channel [recipientlimit](#) (which may be completely overridden on a per-user basis using the LDAP attribute named by the [ldap_recipientlimit](#) MTA option), the TCP/IP-channel-specific option `ALLOW_RECIPIENTS_PER_TRANSACTION` imposes a hard upper-limit, which other settings may only modify by imposing even stricter limits but may not ignore (increase).

49.4.4 TCP/IP-channel-specific options: `ALLOW_REJECTIONS_BEFORE_DEFERRAL` (integer)

This SMTP server option sets a limit on the number of bad (failing) RCPT TO: or VRFY: addresses that will be allowed during a single session. (Note that unlike [ALLOW_RECIPIENTS_PER_TRANSACTION](#) and [REJECT_RECIPIENTS_PER_TRANSACTION](#) where the RCPT TO: and VRFY: counts are separate, this is a *combined* count; failing RCPT TO:'s and VRFY:'s are added together.) That is, after the specified number of To: addresses have been rejected at the RCPT TO: or at VRFY:, all subsequent recipients, good or bad, will be rejected at their RCPT TO: presentations with a temporary error

```
451 4.5.3 Too many rejections; try again later
```

or at VRFY: presentations with a temporary error

```
451 4.5.3 Verification blocked; too many rejections
```

The default is (essentially) no limit. See also the [deferralrejectlimit](#) channel option.

Compare this option with the [REJECT_RECIPIENTS_PER_TRANSACTION](#) TCP/IP-channel-specific option, which will cause *all* recipients to be rejected at the DATA command with a temporary error

```
451 4.5.3 Transaction blocked; too many recipients specified
```

49.4.5 TCP/IP-channel-specific options: `ALLOW_SESSION_BLOCKS` (integer)

The `ALLOW_SESSION_BLOCKS` TCP/IP-channel-specific option for the SMTP server imposes a block limit on the session as a whole. The session is disconnected if this limit is exceeded. If [MTA transaction logging](#) is enabled, the reason field in the close ("X") entry will show:

```
Maximum session data limit of xK octets has been exceeded
```

The default of 0 means no limit is imposed.

49.4.6 TCP/IP-channel-specific options: `ALLOW_TRANSACTION_BLOCKS` (integer)

The `ALLOW_TRANSACTION_BLOCKS` TCP/IP-channel-specific option for the SMTP server imposes a block limit on any transactions. The session is disconnected if this limit is exceeded. If [MTA transaction logging](#) is enabled, the reason field in the close ("X") entry will show:

```
Maximum transaction data limit of xK octets has been exceeded
```

The default of 0 means no limit is imposed.

49.4.7 TCP/IP-channel-specific options: **ALLOW_TRANSACTIONS_PER_SESSION (integer)**

This SMTP/LMTP server option sets a limit on the number of messages allowed per connection. The default is no limit. When the number of messages specified is exceeded, then normally the temporary error

```
451 4.5.3 No more transactions allowed
```

is returned at the MAIL FROM: line; but see the [TRANSACTION_LIMIT_RCPT_TO](#) TCP/IP-channel-specific option which will cause this temporary error to instead be issued at the RCPT TO: line. See also the [transactionlimit](#) and [\(in the case of SMTP\) disconnecttransactionlimit channel options](#), which may be used to set such limits on a per-channel basis (hence providing finer-grained control than the `ALLOW_TRANSACTIONS_PER_SESSION` option which applies to all channels handled by the same default Dispatcher SMTP or LMTP service).

49.4.8 TCP/IP-channel-specific options: **ATTEMPT_TRANSACTIONS_PER_SESSION (integer)**

This SMTP/LMTP client option sets a limit on the number of messages the MTA will attempt to transfer during any one connection session.

49.4.9 TCP/IP-channel-specific options: **AUTH_DEBUG (string)**

(New in Messaging Server 7.4-18.01.) This SMTP server option allow enabling SASL (HULA) debugging. The string argument should consist of one or more, space-separated, of "perf", "connect", "authserv", "hula". In order for this option to take full effect, [slave_debug](#) debugging must be enabled. When this is set, it overrides the [debugkeys](#) setting for SMTP authentication logging purposes.

As an alternative to this channel-level enabling of SASL debug, note the [PORT_ACCESS](#) flag \$A is an alternate way to enable SASL debug on a per-connection basis.

49.4.10 TCP/IP-channel-specific options: **AUTH_PASSWORD (string), AUTH_USERNAME (string), EXTERNAL_IDENTITY (string)**

New in Messaging Server 7.0 update 1. In legacy configuration, these SMTP/LMTP client options provide the credentials for SASL (SMTP AUTH) use by the SMTP/LMTP client; in Unified Configuration, these options are *not* used, having been replaced instead by the [authpassword](#), [authusername](#), and [externalidentity](#) channel options.

ASL authentication will be attempted if either the [maysaslclient](#) or [mustsaslclient](#) [channel option](#) is set, with success required for message transmission if [mustsaslclient](#) is set.

The PLAIN and EXTERNAL SASL mechanisms are currently supported. The [AUTH_USERNAME](#) and [AUTH_PASSWORD](#) TCP/IP-channel-specific options provide the credentials for the plain mechanism and the [EXTERNAL_IDENTITY](#) TCP/IP-channel-specific option provides the identity string for SASL EXTERNAL. ([EXTERNAL_IDENTITY](#) can be set to the empty string to enable SASL EXTERNAL without an identity string.)

49.4.11 TCP/IP-channel-specific options: BANNER_ADDITION (string)

This SMTP/LMTP server option adds the specified string to the SMTP/LMTP banner line. If the length of this addition string plus the MTA's regular banner line string (as constructed normally by the MTA---see in particular the [CUSTOM_VERSION_STRING](#) TCP/IP-channel-specific option) would be more than 80 characters, then the additional text will be put on a continuation line as part of a multi-line banner response, rather than being included on the initial 220 line. If the vertical bar character or "pipe" character, |, is included in the [BANNER_ADDITION](#) string, it is interpreted as a request for a line break; the MTA's regular banner line will be output as the first line of a multi-line banner response, and then each vertical bar separated portion of the string (including the very first, even if it is "short"), will be output on its own, separate line.

49.4.12 TCP/IP-channel-specific options: BANNER_HOST (string)

This option applies to both the SMTP/LMTP server and the SMTP/LMTP client. If specified, this value takes precedence over even a [local_host_alias](#) for the name used in the 220 banner (SMTP/LMTP server) and on the HELO/EHLO/LHLO line (SMTP/LMTP client).

49.4.13 TCP/IP-channel-specific options: BANNER_REVERSE_HOST (boolean)

This option applies to the SMTP/LMTP server. It defaults to 0 (false). If set to 1 (true), when a connection is accepted a reverse DNS lookup is performed on the local host's IP address. if a result is returned it becomes the name used in the server's 220 banner. This value takes precedence over any [BANNER_HOST](#) value as well as any [local_host_alias](#) value.

49.4.14 TCP/IP-channel-specific options: BANNER_PURGE_DELAY (integer)

(New in JES MS 6.3.) 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 start blasting

SMTP commands as soon as the connection is open. 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.

The `BANNER_PURGE_DELAY` SMTP server option exists to implement this strategy. Set the `BANNER_PURGE_DELAY` option to the number of centiseconds to delay before purging the input buffer and sending the banner. The default value of 0 disables both the delay and purge. See also the [PORT_ACCESS mapping table \\$D flag](#), which can also be used to set such a banner purge delay, in a more selective manner. And note that the [disconnectbadcommandlimit channel option](#) may be of interest when using any such banner purge delay technique.

49.4.15 TCP/IP-channel-specific options: BUFFER_SIZE (integer)

(New in JES MS 6.3.) The `BUFFER_SIZE` option is available for the [LMTP server](#) only. It sets the buffer size for the LMTP server to do in-memory buffering of incoming messages, overriding the default 1,000,000 bytes. Thus this LMTP server option is analogous to the [max_internal_blocks](#) MTA option which controls similar buffering for components other than the LMTP server (and has no relationship to the MTA option with a similar name `buffer_size` but a completely different meaning).

49.4.16 TCP/IP-channel-specific options: CHECK_SOURCE (0 or 1)

The MTA's SMTP and LMTP server normally attempts to determine the name of the host from which a connection has been received, as specified by the [ident * channel options](#). When the determined name does not match the name presented by the remote SMTP/LMTP client on the HELO/EHLO/LHLO line, the `CHECK_SOURCE` TCP/IP-channel-specific option controls whether the name found from a DNS lookup (or the IP domain literal, if DNS lookups have been disabled such as with the [identnonnumeric channel option](#)) is included in the constructed Received: header as a comment after the presented name. A value of 1 (the default) enables the inclusion of the determined name when different from the presented name. A value of 0 disables the inclusion of any such comment and thus eliminates one of the more useful checks of message validity. For the SMTP server, setting `CHECK_SOURCE` to 0 also effectively blocks [switchchannel](#) channel switching from taking effect.

49.4.17 TCP/IP-channel-specific options: CLIENT_CERT_NICKNAME (string)

(New in Messaging Server 7.0.5) This SMTP client option specifies the certificate to use for SMTP client authentication via STARTTLS. The nickname follows the same format as other certificate nicknames in Messaging Server. It is split at the first ":" character, and the portion prior to the colon is the security token; the portion subsequent to the ":" character is the certificate nickname in that security token. If no ":" character is present, then the NSS built-in certificate databases are used. Note that `CLIENT_CERT_NICKNAME` should only be used with [IGNORE_BAD_CERT=0](#) set, as otherwise the security it can provide is defeated.

49.4.18 TCP/IP-channel-specific options: COMMAND_RECEIVE_TIME (integer)

This SMTP/LMTP server option specifies, in minutes, how long to wait to receive general SMTP commands, (commands other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

49.4.19 TCP/IP-channel-specific options: COMMAND_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to spend transmitting general SMTP commands, (commands other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

49.4.20 TCP/IP-channel-specific options: CONTINUATION_CHARS (string)

DEPRECATED.

The CONTINUATION_CHARS TCP/IP-channel-specific option is supported by the SMTP server (but not the LMTP server), and by SMTP and LMTP clients. It takes a list of integer values specifying the ASCII positions of additional characters that act like the continuation character specified using the `contchar` channel options; that is, it specifies additional characters that will be accepted as line continuation characters.

This option was used to accomodate the peculiar form of batch SMTP that was employed by BITNET. Now that BITNET is no more, this option is obsolete and deprecated.

49.4.21 TCP/IP-channel-specific options: CUSTOM_VERSION_STRING (string)

Use of this option is **NOT RECOMMENDED!** This SMTP server option (not supported by the LMTP server) sets the IMTA version string to advertise on the SMTP server's 220 banner line. By default, the string "iPlanet Messaging Server <version-number> (<build-date>)" is used. This option is thus a complement to the (also not recommended) [received_version MTA option](#).

49.4.22 TCP/IP-channel-specific options: DATA_RECEIVE_TIME (integer)

This SMTP/LMTP server option specifies, in minutes, how long to wait to receive each line of data during the DATA phase of an SMTP/LMTP dialogue. The default is 5.

49.4.23 TCP/IP-channel-specific options: DATA_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to spend transmitting data during an SMTP/LMTP dialogue. The default is 10.

49.4.24 TCP/IP-channel-specific options: DISABLE_ADDRESS (0 or 1)

The MTA's SMTP server implements a private command XADR. This command returns information about how an address is routed internally by the MTA as well as general channel information. Releasing such information may constitute a breach of security for some sites. Setting the SMTP server option `DISABLE_ADDRESS` to 1 disables the XADR command, so that the SMTP server returns the response

```
550 5.7.0 XADR command has been disabled.
```

to XADR attempts. The default is 0, which enables the XADR command. In 8.0, the default has changed to 1, disabling the XADR command.

SMTP server probes of the [PORT_ACCESS mapping table](#) can enable use of XADR (overriding a more general `DISABLE_ADDRESS=0` setting) for connections from particular source IPs (*e.g.*, the host itself, 127.0.0.1, and particular administrator systems) via the (new in 7.0?) `$V` flag.

49.4.25 TCP/IP-channel-specific options: DISABLE_CIRCUIT (0 or 1)

The MTA's SMTP server implements a private command XCIR. This command returns MTA circuit check information. Releasing such information may constitute a breach of security for some sites. Setting the SMTP server option `DISABLE_CIRCUIT` to 1 disables the XCIR command, so that the SMTP server responds with the error

```
550 5.7.0 XCIR command has been disabled.
```

to any XCIR command attempts. Setting `DISABLE_CIRCUIT` to 0 enables the XCIR command. If `DISABLE_CIRCUIT` is not explicitly set, then use of the XCIR command is controlled by the [DISABLE_GENERAL](#) TCP/IP-channel-specific option setting. In 8.0, the default has changed to 1, disabling the XCIR command.

SMTP server probes of the [PORT_ACCESS mapping table](#) can enable use of XCIR (overriding a more general `DISABLE_ADDRESS=0` or `DISABLE_GENERAL=0` setting) for connections from particular source IPs (*e.g.*, the host itself, 127.0.0.1, and particular administrator systems) via the (new in Messaging Server 7.0) `$V` flag.

49.4.26 TCP/IP-channel-specific options: DISABLE_EXPAND (0 or 1)

The SMTP EXPN command is used to expand mailing lists. Exposing the contents of mailing lists to outside scrutiny may constitute a breach of security for some sites. The SMTP server option `DISABLE_EXPAND`, when set to 1, disables the EXPN command completely, causing the SMTP server to issue the response

```
550 5.7.2 EXPN command has been disabled
```

to any EXPN command. The default value is 0, which causes the EXPN command to work normally.

See also the [expnallow](#), [expndefault](#), and [expndisable](#) channel options that can be used to control this behavior on a per-channel, rather than per-SMTP-server, basis. And see the [expandable_default](#) MTA option which sets an MTA-wide default.

Note that mailing list expansion can also be blocked on a list-by-list basis with the Unified Configuration alias options [alias_expandable](#) and [alias_nonexpandable](#), or the [\[EXPANDABLE\]](#) and [\[NONEXPANDABLE\]](#) alias file named parameters.

Note that an LDAP attribute named by the [ldap_expandable](#) MTA option (normally set to the attributes `mgmanMemberVisibility` and `expandable`) can also be used to disable expansion on a list-by-list basis (and can be used on user entries as well).

49.4.27 TCP/IP-channel-specific options: DISABLE_GENERAL (0 or 1)

The MTA's SMTP server implements a private command XGEN. This command returns status information about whether a compiled configuration and compiled character set are in use. Releasing such information may constitute a breach of security for some sites. Setting the SMTP server option `DISABLE_GENERAL` to 1 disables the XGEN command, so that the MTA responds with

```
550 5.7.0 XGEN command has been disabled.
```

to any XGEN command. The default is 0, which enables the XGEN command. In 8.0, the default has changed to 1, disabling the XGEN command.

SMTP server probes of the [PORT_ACCESS mapping table](#) can enable use of XGEN (overriding a more general `DISABLE_GENERAL=0` setting) for connections from particular source IPs (e.g., the host itself, 127.0.0.1, and particular administrator systems) via the (new in Messaging Server 7.0) `$V` flag.

49.4.28 TCP/IP-channel-specific options: DISABLE_SEND (0 or 1)

This option applies to the MTA's SMTP server. The SMTP server is able to support the `SEND FROM:`, `SAML FROM:`, and `SOML FROM:` SMTP commands (see [RFC 821](#)) that allow for sending a broadcast message to a logged in user on the MTA system rather than or in addition to, an e-mail message. This support is disabled by default (`DISABLE_SEND=1`), so that the MTA will respond with

```
550 5.7.2 SEND/SAML/SOML commands have been disabled.
```

to any such commands. To enable support for these SMTP commands, set `DISABLE_SEND=0`.

49.4.29 TCP/IP-channel-specific options: DISABLE_STATUS (0 or 1)

The MTA's SMTP server implements a private command XSTA. This command returns status information about the number of messages processed and currently in the MTA channel queues, status information about current associations (TCP/IP connections), and status information about LDAP lookup caching. Releasing such information may constitute a breach of security for some sites. Setting the SMTP server option `DISABLE_STATUS` to 1 disables the XSTA command, so that the SMTP server responds with

550 5.7.0 XSTA command has been disabled.

The default is 0, which enables the XSTA command. In 8.0, the default has changed to 1, disabling the XSTA command.

SMTP server probes of the [PORT_ACCESS mapping table](#) can enable use of XSTA (overriding a more general `DISABLE_STATUS=0` setting) for connections from particular source IPs (*e.g.*, the host itself, 127.0.0.1, and particular administrator systems) via the (new in Messaging Server 7.0) `$V` flag.

49.4.30 TCP/IP-channel-specific options: DOT_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to spend transmitting the dot (period) terminating the data in an SMTP/LMTP dialogue. The default is 10.

49.4.31 TCP/IP-channel-specific options: FAST_SMTP_SESSION_TIME_LIMIT (integer)

This SMTP server option controls the time threshold for whether or not to create a `*.data-failed` file in cases where the SMTP connection has aborted (for instance, due to an SMTP client timing out) after the MTA SMTP server has received the final "." terminating the data, but before the MTA has acknowledged receipt with a "250 2.5.0 Ok" message. If the transaction (message transfer) has taken more than this amount of time, in seconds, then normally (see the [REUSE_TIMED_OUT_TRANSFERS](#) TCP/IP-channel-specific option) a `*.data-failed` file will be created in such cases where the MTA has not been able to successfully send back its acknowledgement of receipt of the message. If a transaction has taken less than the specified amount of time, in seconds, then a `*.data-failed` file will not be created. The default is 30.

49.4.32 TCP/IP-channel-specific options: HELLO_RECEIVE_TIME (integer)

(New in Messaging Server 7.0-3.01.) This SMTP server (but not LMTP server) option controls how long, in minutes, the SMTP server will wait for a client's initial HELO or EHLO command. (Prior to the addition of this option, that timeout was controlled by the [COMMAND_RECEIVE_TIME](#) TCP/IP-channel-specific option, which still controls the SMTP server's timeout for other commands.) The default is 1 (minute).

49.4.33 TCP/IP-channel-specific options: HIDE_VERIFY (0 or 1)

The SMTP VRFY command can be used to establish the legality of an address prior to actually using it. Unfortunately this command has been abused by automated query engines in some cases. The SMTP server option `HIDE_VERIFY`, when set to 1, tells the MTA not to return any useful information in the VRFY command result; in particular, when `HIDE_VERIFY=1` is set the MTA will, for arguments that are at least syntactically legal, return for addresses that syntactically might be "local" only:

252 2.5.0 Possible local address <address>

or return the same response it returns regardless of HIDE_VERIFY setting for apparently remote addresses, namely:

252 2.5.0 Possible remote address not checked.

The default value is 0, which causes VRFY to act normally. See also the channel options controlling this behavior, [vrfyallow](#), [vrfydefault](#), and [vrfyhide](#).

49.4.34 TCP/IP-channel-specific options: IGNORE_BAD_CERT (bit-encoded integer)

This SMTP client and SMTP server option controls the MTA's reaction to bad (remote) certificates. Bit 0 (value 1) controls whether a bad client certificate is ignored even when [musttls](#) or [musttlsserver](#) is set. Bit 1 (value 2) controls whether a bad server certificate is ignored even when [musttls](#) or [musttlsclient](#) is set. (Note that bad certificates are always allowed in other cases -- e.g., when [maytls*](#) is set.) The default is 3.

49.4.35 TCP/IP-channel-specific options: INITIAL_COMMAND (string)

The SMTP client option INITIAL_COMMAND may be used to specify an initial command to send, at the beginning of each SMTP session. By default, it is not set.

49.4.36 TCP/IP-channel-specific options: LOG_BANNER (0 or 1)

The SMTP/LMTP client option LOG_BANNER controls whether the remote SMTP/LMTP server banner line is included in [mail.log* file entries](#) when the [logging](#) channel option is enabled for the channel. A value of 1 (the default) enables logging of the remote SMTP server banner line; a value of 0 disables it. LOG_BANNER also affects whether a remote SMTP/LMTP banner line, if available, is included in [bounce messages](#) generated by the channel.

49.4.37 TCP/IP-channel-specific options: LOG_CONNECTION (integer)

The LOG_CONNECTION TCP/IP-channel-specific option controls whether or not connection information, e.g., the domain name of the SMTP client sending the message, is saved in [mail.log file entries](#) and the writing of [connection records](#) when the [logging](#) channel option is enabled for the channel; it applies to both SMTP server and SMTP client, and to both LMTP server and LMTP client. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Table 49.2 LOG_CONNECTION TCP/IP-channel-specific option bit mask values

Bit	Value	Usage
-----	-------	-------

0	1	When set, connection information is included in E and D log records.
1	2	When set, connection open/close/fail records are logged by message enqueue and dequeue agents such as the SMTP and X.400 clients and servers.
2	4	When set, I records are logged recording ETRN events.

Bit 0 is the least significant bit.

This TCP/IP-channel-specific option defaults to the setting of the general MTA option [log_connection](#) (as set in the MTA option file (legacy configuration) or at the `mta` level in Unified Configuration). The TCP/IP-channel-specific option may be set explicitly to override on a per-channel basis the behavior requested by the general option. That is, a pair of settings such as:

```
msconfig> set mta.log_connection 7
msconfig> set channel:tcp_submit.options.LOG_CONNECTION 0
```

means that arbitrary `tcp_*` channels will log connection information, except for the `tcp_submit` channel which will *not* log such connection information.

49.4.38 TCP/IP-channel-specific options: LOG_TRANSPORTINFO (0 or 1)

The SMTP/LMTP client option `LOG_TRANSPORTINFO` controls whether or not transport information, such as the sending and receiving side IP numbers and port numbers:

```
TCP | MTA-IP | MTA-port | remote-IP | remote-port
```

are included in [mail.log file dequeue entries](#) when the [logging](#) channel option is enabled for the channel. A value of 1 enables transport information logging. A value of 0 disables it. If not explicitly set, the `LOG_TRANSPORTINFO` TCP/IP-channel-specific option defaults to 1 (transport information logging enabled) if at least one of bits 0 or 3 (values 1 or 8) is set for the MTA option [log_connection](#) (`log_connection` as set in the MTA option file in legacy configuration, or at the `mta` level in Unified Configuration), and otherwise `LOG_TRANSPORTINFO` defaults to 0. The `LOG_TRANSPORTINFO` TCP/IP-channel-specific option may be set explicitly for a channel to control the logging of transport information regardless of whether connection logging is generally enabled.

`LOG_TRANSPORTINFO` also affects whether transport information, if available, is included in [bounce messages](#) generated by the channel.

49.4.39 TCP/IP-channel-specific options: MAIL_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to spend transmitting the SMTP command `MAIL FROM:`. The default is 10.

49.4.40 TCP/IP-channel-specific options: MAILBOX_BUSY_FAST_RETRY (integer)

(New in JES MS 6.3.) This LMTP client option controls whether the [LMTP client](#) requests that the [Job Controller](#) schedule the message's next retry attempt with a special "short delay" (overriding the channel's usual [backoff](#) setting) in cases of temporary errors with a 4.2.1 extended response code. Since 4.2.1 is the extended response code used for the "Mailbox is busy" (IMAP_MAILBOX_LOCKED) error condition, which can occur when a mailbox is locked by another process due to another message delivery or to an IMAP message operation, in such cases attempting another delivery "quickly" is likely to be useful.

The default is 1, meaning to ask the Job Controller for a fairly quick retry. Setting the option to 0 disables the override of [backoff](#). Positive values greater than 1 result in a retry, but not quite as quick of one. The default value of 1 corresponds to the (not configurable) formula that the [ims-ms channel](#) uses for such retries. (In 6.3 and 7.0.5.32 and later, the formula used is $n + \text{rand}() \% (120 * n)$ seconds. In release 7.0 through 7.0.5.31.0, the formula used is $n + \text{rand}() \% (4 * n)$ seconds.)

49.4.41 TCP/IP-channel-specific options: MAX_A_RECORDS (integer)

This SMTP/LMTP client option specifies the maximum number of A records that the MTA should try using when attempting to deliver a message. The default is no limit.

49.4.42 TCP/IP-channel-specific options: MAX_B_ENTRIES (integer)

New in JES MS 6.2. This TCP/IP-channel-specific SMTP server option specifies the maximum number of "B" entries per SMTP connection session that the server will write to the [MTA message transaction log \(mail.log*\) file](#). Bad commands sent to the SMTP server are logged as "B" records in the MTA's message log file (mail.log*), if message logging has been enabled (the [logging](#) channel option). In such "B" entries, the recipient address field will contain the bad command that was rejected by the SMTP server as invalid, while the diagnostic field will contain the response the SMTP server gave. Once the value of `MAX_B_ENTRIES` has been reached, the SMTP server writes its final (for that session) "B" record, with the string

```
(limit reached; last B record for this session)
```

appended to the diagnostic field. The default value for `MAX_B_ENTRIES` is 10. Note that keeping this value set to a reasonable size keeps the MTA from wasting too much time writing B entries for unrecognized/illegal command attempts, so that deliberate fake command attempts will not distract the MTA from more important work; *i.e.*, it protects against a certain form of "denial of service" attack. See also the (new in JES MS 6.2) [disconnectbadcommandlimit](#) channel option.

49.4.43 TCP/IP-channel-specific options: MAX_CLIENT_THREADS (integer)

This SMTP/LMTP client option takes an integer number indicating the maximum number of simultaneous, outbound connections that the client channel program will allow. Note that multiple processes may be used for outbound connections, depending on how you have [channel processing pools](#) set up. This TCP/IP-channel-specific option controls the number of threads per process. The default value is 10.

49.4.44 TCP/IP-channel-specific options: MAX_HELO_DOMAIN_LENGTH (integer)

This SMTP/LMTP server option may be used to specify a maximum length of host name that a remote client may put on its HELO or EHLO line. If a longer name is seen on the HELO or EHLO line, then the SMTP or LMTP server will reject the command with an error (in the case of the SMTP server) of

```
501 5.5.0 Argument to HELO is too long.
```

or

```
501 5.5.0 Argument to EHLO is too long.
```

or (in the case of the LMTP server) of

```
501 5.5.0 Argument to LHLO is too long.
```

as appropriate. The default is no limit.

49.4.45 TCP/IP-channel-specific options: MAX_J_ENTRIES (integer)

This TCP/IP channel SMTP server option specifies the maximum number of "J" [mail.log* entries](#) to write during a single SMTP connection session. (Prior to JES MS 6.2 it also limited how many increments of the ["Rejected" counter](#) would occur during a single transaction.) The default is 10.

Note that keeping this value set to a reasonable size keeps the MTA from wasting too much time writing "J" entries for unsuccessful submission attempts, so that deliberate unsuccessful submission attempts will not distract the MTA from more important work; *i.e.*, it protects against a certain form of "denial of service" attack.

When this limit is reached, the final "J" record for the session will contain (in addition to the "regular" text in a "J" record) the extra text "(limit reached; last J record for this session)". (As of JES MS 6.2, the "Rejected" counter will continue to be incremented for each additional rejected attempt during the session.)

49.4.46 TCP/IP-channel-specific options: MAX_MX_RECORDS (integer <= 32)

This TCP/IP channel SMTP/LMTP client option specifies the maximum number of MX records that the MTA should try using when attempting to deliver a message. The maximum value is 32, which is also the default. (Note that the [IP_ACCESS](#) mapping table provides an alternate means to achieve such limits.)

49.4.47 TCP/IP-channel-specific options: MAX_SERVER_THREADS (integer; <= 47 on Solaris)

This TCP/IP channel SMTP/LMTP server option has little relevance today, and the little effect it does have is Solaris-specific. (Formerly, for the PMDF, pre-Dispatcher multithreaded SMTP server, it controlled the maximum number of simultaneous, inbound connections that the pre-Dispatcher multithreaded SMTP server program would allow. Note that since only one such server process was allowed, this option effectively controlled the total number of simultaneous, inbound SMTP connections that older versions of PMDF could handle.) Nowadays, this option's sole effect is that on Solaris only, it is taken into account in deciding the maximum number of file descriptors allowed for an SMTP server process. Since this value is multiplied by 5 and then 20 added, the effect of the default of 40 is that a maximum of 220 file descriptors are allowed.

The default value is 40. Note that setting a value of 48 or more on Solaris will cause the server process to exit with an error.

49.4.48 TCP/IP-channel-specific options: OPEN_CONNECTION_TIME (integer)

RESTRICTED. Not yet implemented (due to issues with TCP/IP package support).

49.4.49 TCP/IP-channel-specific options: PACKET_SIZE_LIMIT (integer)

RESTRICTED. This option exists for potential debugging purposes; it should not normally be set by sites.

The `PACKET_SIZE_LIMIT` TCP/IP-channel-specific option is available both for SMTP and for LMTP. The default is 4096. Attempts to set this option to more than 4096 will result in a value of 4096 being used.

49.4.50 TCP/IP-channel-specific options: PROXY_PASSWORD (string)

The legacy configuration TCP/IP-channel-specific option `PROXY_PASSWORD` has been replaced in Unified Configuration by the [smtpproxypassword](#) MTA option.

In legacy configuration, in order for the MMP to perform SMTP proxying, this SMTP server option must be set to the same "secret" password as the MMP's [SmtProxyPassword](#) value. If the `PROXY_PASSWORD` TCP/IP-channel-specific option is not set, client attempts to use the XPEHLO command will receive an error:

```
503 5.5.0 Proxy support is not enabled.
```

If the `PROXY_PASSWORD` TCP/IP-channel-specific option is set but its value does not match the MMP's value, client attempts to use the XPEHLO command will receive an error:

```
535 5.7.8 SMTP proxy authentication check failed.
```

49.4.51 TCP/IP-channel-specific options: RCPT_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to spend transmitting the SMTP command RCPT TO:. The default is 10.

49.4.52 TCP/IP-channel-specific options: REJECT_RECIPIENTS_PER_TRANSACTION (integer)

This SMTP server option may be used to specify a limit on the number of recipients that will be accepted during a single transaction. For the SMTP server, it also limits the number of VRFY address verifications that may be performed. (Note that the count of actual recipients, RCPT TO:, is separate from the count of verifies, VRFY:; that is, VRFY:'s do not count against the RCPT TO: limit, nor do RCPT TO:'s count against the VRFY: limit; each is limited *independently* to the REJECT_RECIPIENTS_PER_TRANSACTION value.) If the RCPT TO: limit is exceeded, then at the DATA command the entire message to *all* recipients will be rejected with a temporary error:

```
451 4.5.3 Transaction blocked; too many recipients specified
```

(Compare with [ALLOW_RECIPIENTS_PER_TRANSACTION](#) which rejects merely the excess recipients with a

```
451 4.5.3 Too many recipients specified
```

error at the RCPT TO: command, allowing the message to be submitted to the initial recipients.) Attempts to VRFY more addresses than the limit will be rejected with a

```
451 4.5.3 Verification blocked; too many operations performed
```

error. The default is no limit. Note that if both `ALLOW_RECIPIENTS_PER_TRANSACTION` and `REJECT_RECIPIENTS_PER_TRANSACTION` are set, with `REJECT_RECIPIENTS_PER_TRANSACTION` being set to a larger value than `ALLOW_RECIPIENTS_PER_TRANSACTION`, then once `ALLOW_RECIPIENTS_PER_TRANSACTION` is exceeded any additional recipients receive a temporary error, and once `REJECT_RECIPIENTS_PER_TRANSACTION` is exceeded then the entire message is rejected with a temporary error.

See also the [recipientcutoff](#) and [disconnectrejectlimit](#) channel options, and the [ldap_recipientcutoff](#) and [ldap_domain_attr_recipientcutoff](#) MTA options.

49.4.53 TCP/IP-channel-specific options: REUSE_TIMED_OUT_TRANSFERS (0 or 1)

This SMTP server option controls creation and use of `*.data-failed` files; the default is 1, meaning that such files may be created and used. More specifically, this option controls whether the MTA writes and uses `IMTA_QUEUE:tcp_*/spool/*.data-failed` files (JES MS 6.3 and earlier) or in JS Messaging Server 7.0, `$SERVERROOT/data/queue/tcp_*/spool/*.data-failed` files. If enabled, the MTA writes such a file in cases where an SMTP client connection is dropped after the MTA has received the final "." terminating the message data, but before the MTA has successfully sent its "250 2.5.0 Ok." message acknowledging the receipt of the message back to the client, and checks for the existence of such a file when receiving an incoming message (so as to detect a "duplicate" attempted message submission). When such files are being created/used, the MTA will do a hash of each incoming message to compare the hash against any stored `*.data-failed` files to determine whether the current incoming message had in fact already been received in a previous transaction. When the MTA does see such a case, the MTA will issue a

```
250 2.5.0 Prior aborted transfer used
```

success back to the sending client (after the client finishes sending the message data). The MTA will only create such a `*.data-failed` file in the case of an incoming message where there was a significant delay in our sending of the "250 ok"; see the [FAST_SMTP_SESSION_TIME_LIMIT](#) TCP/IP-channel-specific option. The MTA assumes that cases of "short" disconnect times are where it's just a poorly behaved client, one that likes to disconnect immediately without waiting for the acknowledgement, that won't in fact be resending the message. Whereas the "long" timeout case, where the MTA will still make the `*.data-failed` file, is a case where it's reasonable that the client timed out on the connection, and the client likely will feel a need to try resending the message.

The `*.data-failed` files are normally retained in the incoming TCP/IP channel's `spool` subdirectory for the number of days specified by an `IMTA_TCP_FLAG_RETENTION` Tailor option, which defaults to 7 if not explicitly set. (The `return_units` MTA option has no effect here; this Tailor option is always interpreted in units of days.) Thus by default (no `IMTA_TCP_FLAG_RETENTION` Tailor option set) such `*.data-failed` files are retained for seven days. Note that disabling the MTA's SMTP server creation and use of `*.data-failed` files by setting `REUSE_TIMED_OUT_TRANSFERS=0` can provide a noticeable performance (throughput) increase---perhaps 30% for the SMTP server; the downside is that disabling their use means that the MTA will no longer detect and avoid certain cases of duplicate submissions of messages, so users may receive "duplicate" copies of messages that might have been avoided.

49.4.54 TCP/IP-channel-specific options: SESSION_TIME (integer)

(New in Messaging Server 7.0-3.01.) This SMTP server (but not LMTP server) option controls the maximum amount of time, in minutes, that an SMTP session is allowed before the session will be disconnected. The default is 1200 (minutes). Note that, for performance reasons, this value is checked only once every 10 reads (in order to save on `time()` calls). Once the SMTP server notices that the specified session time limit has been exceeded, the session will be disconnected with an SMTP error:

```
450 4.7.0 Maximum session time of n minutes has been exceeded
```

49.4.55 TCP/IP-channel-specific options: SIZE_DELAY_THRESHHOLDS, SIZE_DELAY_AMOUNTS, RECIPIENT_DELAY_THRESHHOLDS, RECIPIENT_DELAY_AMOUNTS, TRANSACTION_DELAY_THRESHHOLDS, TRANSACTION_DELAY_AMOUNTS (comma-separated list of integers)

These are SMTP server options; (they are also supported by the LMTP servers, though unlikely to be useful in an LMTP context). The number of total message size, number of recipients, and number of transactions are tracked by the SMTP server on a per-session basis. And via these options, the SMTP server can impose punitive delays in responding when various threshold levels are exceeded. That is, the SMTP server can in graduated fashion "slow down" its responses to clients that are submitting "large" numbers of "large" messages to "large" numbers of recipients.

These options come in pairs, each containing a list of up to ten, comma-separated integers. (The default is all zeros.) One list is the threshold list, which specifies the point past which the additional delay specified by the corresponding number in the other list applies. The delay amounts are additive; that is, each delay value specifies an additional delay to impose, rather than representing an absolute delay. The current delay in effect is maximized with the result of summing the three contributors:

$$D_c = \max(D_c, D_t + D_r + D_s)$$

Here D_c is the current delay, D_t is the delay due to transactions being exceeded, D_r is the delay due to recipients being exceeded, and D_s is the delay due to size being exceeded.

Note that the `SIZE_DELAY_THRESHHOLDS` are in bytes, *not* blocks. The `*_DELAY_AMOUNTS` values are interpreted as hundredths of seconds.

See also [MeterMaid](#) for an alternate way of implementing such intentional response delays.

49.4.56 TCP/IP-channel-specific options: SSL_CLIENT (0 or 1)

(New in 7.0.5) If set to 1, SSL/TLS negotiation will be performed immediately after any connection is established by the SMTP client (without any use of STARTTLS). It is thus the client analogue of the Dispatcher's `ssl_ports` option (`TLS_PORT` Dispatcher option in legacy configuration) for servers. The default for `SSL_CLIENT` is 0, meaning that such automatic (non STARTTLS) SSL/TLS negotiation will not be done by the client. (Client negotiation via STARTTLS is a separate issue and may still be performed, depending upon configuration; see the [*tlsclient channel options](#).)

`SSL_CLIENT` is useful for establishing point to point links to other systems using `smtps`: on port 465. In particular, a typical use of the `SSL_CLIENT` option would be on a special `tcp_*` channel that has `port 465` set; e.g.:

```
msconfig> show channel:tcp_ssl_friend.port
instance.channel:tcp_ssl_friend.port = 465
msconfig> show channel:tcp_ssl_friend.options.SSL_CLIENT
instance.channel:tcp_ssl_friend.options.SSL_CLIENT = 1
```

49.4.57 TCP/IP-channel-specific options: STARTTLS_FAILURE_RECONNECT_DELAY (integer)

(New in 7.0.5.32.) When [maytlsclient](#) is in effect, this SMTP/LMTP client option specifies, in centiseconds, how long to wait after a TLS negotiation failure before reattempting connection without use of TLS. The default is 200.

49.4.58 TCP/IP-channel-specific options: STATUS_DATA_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to wait to receive the SMTP response to our sent data; *i.e.*, how long to wait to receive a "250" (or other) response to the dot terminating sent data. The default value is 10.

See also the [STATUS_DATA_RECV_PER_ADDR_TIME](#), [STATUS_DATA_RECV_PER_BLOCK_TIME](#), and [STATUS_DATA_RECV_PER_ADDR_PER_BLK_TIME](#) TCP/IP-channel-specific options.

49.4.59 TCP/IP-channel-specific options: STATUS_DATA_RECV_PER_ADDR_TIME (floating point value)

This SMTP/LMTP client option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of addresses in the MAIL TO: command. This value is multiplied by the number of addresses and added to the base wait time (specified with the [STATUS_DATA_RECEIVE_TIME](#) TCP/IP-channel-specific option). The default is 0.083333 (in units of minutes per address).

49.4.60 TCP/IP-channel-specific options: STATUS_DATA_RECV_PER_BLOCK_TIME (floating point value)

This SMTP/LMTP client option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of blocks sent. This value is multiplied by the number of blocks and added to the base wait time (specified with the [STATUS_DATA_RECEIVE_TIME](#) TCP/IP-channel-specific option). The default is 0.001666 (in units of minutes per block). (For purposes of this option, a block is always 512 bytes, *not* whatever block size might be defined by the [block_size](#) MTA option.)

49.4.61 TCP/IP-channel-specific options: STATUS_DATA_RECV_PER_ADDR_PER_BLK_TIME (floating point value)

This SMTP/LMTP client option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of addresses (in the MAIL TO: command) per number of blocks sent. This value is multiplied by the number of addresses per block and added to the base wait time (specified with the [STATUS_DATA_RECEIVE_TIME](#) TCP/IP-channel-specific option). The default is 0.003333 (in units of minutes per block per address). (For purposes of this option, a block is always 512 bytes, *not* whatever block size might be defined by the [block_size](#) MTA option.)

49.4.62 TCP/IP-channel-specific options: STATUS_MAIL_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to wait to receive the initial 220 banner line, how long to wait to receive a response to an HELO, EHLO, or RSET command, and how long to wait to receive the SMTP response to a sent MAIL FROM: command. The default is 10.

49.4.63 TCP/IP-channel-specific options: STATUS_RCPT_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to wait to receive the SMTP response to a sent RCPT TO: command. The default value is 10.

49.4.64 TCP/IP-channel-specific options: STATUS_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies, in minutes, how long to wait to receive the SMTP reply to general SMTP commands, (replies other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

49.4.65 TCP/IP-channel-specific options: STATUS_TRANSMIT_TIME (integer)

This SMTP/LMTP server option specifies, in minutes, how long to spend transmitting the SMTP/LMTP reply to an SMTP/LMTP command. The default value is 10.

49.4.66 TCP/IP-channel-specific options: TRACE_LEVEL (0, 1, or 2)

This TCP/IP channel SMTP server and SMTP client option, and LMTP server and LMTP client option, controls whether TCP/IP level trace is included in debug log files. The default value is 0, meaning that no TCP/IP packet traces are included; a value of 1 tells the MTA to include TCP/IP packet traces in any debug log files; a value of 2 tells the MTA to include some additional information, such as DNS lookup information, in addition to the basic TCP/IP packet traces.

49.4.67 TCP/IP-channel-specific options: TRANSACTION_LIMIT_RCPT_TO (0 or 1)

This SMTP/LMTP server option affects the MTA's behavior once [ALLOW_TRANSACTIONS_PER_SESSION](#) has been exceeded. The default is 0, meaning that once [ALLOW_TRANSACTIONS_PER_SESSION](#) has been exceeded, then the MTA will reject subsequent transactions (during that same session) at the MAIL FROM: command. If set to 1, the subsequent transactions will instead be rejected at the RCPT TO: command. In either case, the rejection will be done via an error:

```
451 4.5.3 No more transactions allowed
```

49.4.68 TCP/IP-channel-specific options: TRANSACTION_TIME (integer)

(New in Messaging Server 7.0-3.01.) This SMTP server (but not LMTP server) option controls the maximum amount of time, in minutes, that an SMTP transaction is allowed before the session will be disconnected. The default is 600 (minutes). Note that, for performance reasons, this value is checked only once every 10 reads (in order to save on time() calls). Once the SMTP server notices that the specified transaction time limit has been exceeded, the session will be disconnected with an SMTP error:

```
450 4.7.0 Maximum transaction time of n minutes has been exceeded
```

49.4.69 TCP/IP-channel-specific options: WINDDOWN_TIMEOUT (integer)

This SMTP/LMTP client option sets the time, in seconds, to wait before killing active delivery threads. The default is 60*90 seconds (*i.e.*, 90 minutes).

49.5 AUTH_ACCESS mapping table

New in 7.0.5 is the [AUTH_ACCESS](#) mapping table, which has the purpose of SMTP session control. This access mapping is consulted during SMTP/LMTP client operations just prior to initiating client connections, and in particular, prior to DNS or host table lookup of destination domains. (In particular, and as compared to the later [IP_ACCESS mapping](#), [AUTH_ACCESS](#) is consulted prior to MX record lookups.) The [AUTH_ACCESS](#) mapping allows control or override of various SMTP session level features.

The probe for the [AUTH_ACCESS](#) mapping is of the form:

```
channel | filename | queue-time | envelope-from | auth-parameter | username | domain
```

where *channel* is the channel from which the message is being dequeued, *filename* is the full file name of the message, *queue-time* is the approximate time in seconds that the message has been in the queue (or -1 if that value cannot be determined), *envelope-from* is the envelope From address for this message, *auth-parameter* is the unencoded value of any MAIL FROM AUTH= parameter associated with the message, *username* is the authentication identity used to submit the message (plus an asterisk character suffix), and *domain* is the DNS name of the server to which the message would (if not overridden by this mapping) be sent. Note that the *username* is not necessarily the same as the user's canonical e-mail address (*mail* attribute value) or any other LDAP attribute value; rather, the *username* is a canonical identity constructed as *uid@canonical-domain*. Note also that

the [mapping_paranoid](#) MTA option, if set, will cause any vertical bar characters that would have been in the `envelope-from`, `auth-parameter`, `username`, or `domain` fields to be replaced by the specified character.

The mapping template (right hand side) can contain a number of flags plus a series of vertical bar separated fields. Setting a flag causes the consumption of zero or more fields, processed in the order, and with the effects, shown in [Table of AUTH_ACCESS mapping flags](#).

Table 49.3 AUTH_ACCESS mapping flags

Flag	Fields	Description
\$U		Enable SMTP client debugging for this transaction. No fields are consumed.
\$N	<code>error-text</code>	Abort this connection attempt. One field is used to specify the error text.
\$F	<code>env-from</code>	Override the message's envelope From address. One field is used to specify that address.
\$A	<code>auth-param</code>	Override the MAIL FROM AUTH= parameter. One field is used to specify the override value.
\$Q	<code>authorization username password</code>	Override the credentials used for PLAIN authentication. Three fields are consumed with use of this flag: the first specifies an authorization identity (normally left blank), the second specifies an authentication identity (the "username"), and the third specifies a password.
\$/		Treat a PLAIN authentication attempt failure as an SMTP temporary failure. (Normally, when this flag is not set, such an authentication failure is instead treated as: ignored when maysaslclient is in effect, <i>vs.</i> causing an SMTP permanent failure (bounce) when mustsaslclient is in effect.)
\$D	<code>host</code>	Override the DNS name of the server to which the message will be sent. One field is used to specify the new DNS name. (Note that use of this flag causes the TCP/IP-channel-specific option SSL_CLIENT to be ignored; instead use <code>\$S</code> to enable <code>smtps: use</code> .)
\$P	<code>port</code>	Override the value of the <code>port</code> channel option (default 25). One field is used to specify the new port number. (Note that use of this flag causes the TCP/IP-channel-specific option SSL_CLIENT to be ignored; instead use <code>\$S</code> to enable <code>smtps: use</code> .)
\$S		Enables the use of TLS (<code>smtps:</code>). No fields are consumed. Note that this flag enables <code>smtps: use</code> even when <code>\$D</code> or <code>\$P</code> has also been used.

\$X		Disable MX lookups for connection establishment (overriding any channel <code>*mx</code> channel option setting). No fields are consumed.
\$M		Enable MX lookups for connection establishment (overriding any channel <code>*mx</code> keyword setting); the effect is <code>randommx</code> MX record use. No fields are consumed.
\$B	<code>lastresort-server</code>	Set a <code>lastresort</code> value, overriding the value specified by the <code>lastresort</code> channel option. One field is consumed as the <code>lastresort</code> server name.
\$T		Force <code>musttlsclient</code> on for this message. No fields are consumed.
\$H		Disable TLS for this message. No fields are consumed.
\$G		Force <code>mustsasclient</code> on for this message. No fields are consumed.
\$I		Disable SASL for this message. No fields are consumed.
\$Y		Perform no special overrides and send message normally. This explicit "no-op" result is useful for specifying mapping table match cases that cause SMTP client processing to proceed normally. No fields are consumed.
Input flag comparisons	Fields	Description
\$:		Match only if external material (<i>e.g.</i> , the envelope From address) in the probe contained a vertical bar
\$;		Match only if no vertical bars were present in any external material in the probe

The AUTH_ACCESS mapping table can be used for a variety of purposes, including various special-purpose, targeted, override effects on SMTP connections. However, the combination of effects it allows is especially intended to facilitate special-purpose identity/authentication scenarios, such as effective "on behalf of submission", also called "third party submission".

For instance, suppose that local user `adam.brown@local.domain.com` also has a remote identity and mailbox as `abrown@remote.domain.com`, and that this local user when submitting messages through your MTA would sometimes, for some messages, like those message to go out under the remote identity/address. In the example below, for specificity, assume further that the user client will authenticate when submitting such messages, submitting with the user's normal, local address as the envelope From, but with a MAIL FROM AUTH= parameter set to the remote address. Then an AUTH_ACCESS mapping to redirect such messages to a remote.domain.com server and submit the messages using the remote identity could be:

```
tcp_local|*|*|adam.brown@local.domain.com|abrown@remote.domain.com|*$adam.brown@local.domain.com|* \
$Fabrown@remote.domain.com|$Q|abrown@remote.domain.com|remotepassword|$/$Dremote.domain.com|$P587
```

If such a setup is desired for multiple users, rather than just one or a few special users hard-coded (with their remote passwords!) into the AUTH_ACCESS mapping, then a more real-world example might also include storing such users' remote credentials (remote identity and remote password) in some data repository, for instance, perhaps in the regular user LDAP directory, or perhaps in a special LDAP directory accessed via extldap: URLs, or even in some other database, and then looking up the addresses and credential data when messages come through with a MAIL FROM AUTH= parameter differing from the envelope From. Note that in such setups, one of the most challenging aspects (not from the MTA configuration point of view, but rather from the design and maintenance of the data point of view) is likely to be establishing, and maintaining, a tight correspondence between each such local user identity and the remote identity (or identities) that local user is authorized to use.

In [Example 18-3](#), when SMTP AUTH was used to submit a message so that a username is present for the message, if also a MAIL FROM AUTH= parameter is present and differs from the username, then the username is looked up in LDAP and that user's LDAP entry is checked for whether a value of a special LDAP attribute, here assumed to be mailRemoteIdentity, matches the MAIL FROM AUTH= parameter value.

The user data is assumed, for purposes of [Example 18-3](#), to be organized in two LDAP directories: the usual user/group LDAP directory (containing, in addition to all the usual attributes for users, a special site-added, potentially multi-valued, mailRemoteIdentity attribute, used to store any remote addresses that user is permitted to use), as well as a so-called "external" LDAP directory (containing remote domain names under which are stored the remote addresses with their credentials and an attribute for each remote address specifying which local address(es) are permitted to use that remote identity). See [Example 18-1](#) and [Example 18-2](#) for example excerpts of such a data setup. Administratively, the management and updating and access to the data in the "external" LDAP directory may well be somewhat separate and different than for the usual user/group LDAP directory. See the MTA options for configuration of external LDAP lookups, discussed in [LDAP external directory lookup MTA options](#).

49.5.1 AUTH_ACCESS mapping example: Excerpt of local user entry in user/group LDAP

In the local.domain.com users portion of the DIT:

```
mail: adam.brown@local.domain.com
mailRemoteIdentity: abrown@remotel.domain.com
mailRemoteIdentity: adam.brown@remote2.domain.com
```

49.5.2 AUTH_ACCESS mapping example: Excerpt of remote identity entries in alternate (external) LDAP

Under the remotel.domain.com portion of the "external" LDAP DIT:

```
mail: abrown@remotel.domain.com
username: remotel-username
```

```
password: remote1-password
submitter: adam.brown@local.domain.com
```

Under the remote2.domain.com portion of the "external" LDAP DIT:

```
mail: adam.brown@remote2.domain.com
username: remote2-username
password: remote2-password
submitter: adam.brown@local.domain.com
```

Another important aspect to consider in such setups is error handling: what should happen to messages when (and note that it is almost certain to be a "when" not merely an "if" occurrence) the remote credentials are not accepted by the remote server, or the remote SMTP SUBMIT server is unavailable for an extended period of time. One possible approach, though surely not the only approach, is to have the MTA repeat attempting the remote submission a few times, but then "fall back" to emitting the message instead with the original From address (the locally verified, local user identity) as the sender. The probe to AUTH_ACCESS has access to the message filename and queue-time, which allows differential behavior based on the name (hence number of delivery attempts) or age (time in queue) of messages. And since AUTH_ACCESS operates by optionally overriding for a delivery attempt (but not in the underlying message file on disk!) delivery attempt aspects such as envelope From address, credentials, and remote server (and port) to which to connect, then message aspects such as original envelope From, and recipient destination domain remain present in a message file that has failed delivery attempts, still available for "normal" use should one wish the MTA to "fall back" to attempting a normal delivery without regard to the purported remote identity. [Example 18-3](#) incorporates such checks both on the number of delivery attempts, as well as the age (time in queue) of a message, in order to "fall back" to "normal" delivery (stop attempting the remote identity submission) after some elapsed time and number of attempts.

49.5.3 AUTH_ACCESS mapping example: third party submission

AUTH_ACCESS

```
! The following three entries detect the three cases, respectively:
!   (1) no MAIL FROM AUTH= parameter
!   (2) no username (message submitted without SMTP AUTH use)
!   (3) MAIL FROM AUTH= parameter matches username
! These are three cases where DUE TO INHERENT FEATURES of the original
! message submission, the message will be sent "normally" (no
! special action taken).
!
tcp_local|*|*|*|*|*          $Y
tcp_local|*|*|*|*|*          $Y
tcp_local|*|*|*|*|$3*|*      $Y
!
! Now at the case where the MAIL FROM AUTH= parameter differs from the username.
!
! The following entry uses the subsidiary mapping table X-FILE-IS-OLD to
! check whether the message is "old" either in terms of retries (has had three
! or more delivery attempts) or in terms of time-in-queue (has been in the MTA
! queue for more than 3 hours). If the message file is "old" in either sense,
! presumably due to trouble performing the remote identity submission,
! then "fall back" to sending the message "normally" (no further remote
! identity submission attempts). That is, detect the case of a message where
! third party submission seemed appropriate and was attempted, but DUE TO
! OPERATIONAL TROUBLE with the third party submission, it is now desired to
```

Third party submission example

```
! "fall back" to "normal" delivery.
!
tcp_local|*|*|*|*|*|*   $C$|X-FILE-IS-OLD;$0$|$1|$Y
!
! If the X-FILE-IS-OLD mapping check "failed" (the message file is still fresh),
! then fall through (continue) with the same input probe.
! So look up the authenticated sender identity in the user LDAP directory and
! check a special mailRemoteIdentity attribute to see whether that sender
! should be allowed to try sending with that AUTH= parameter.
!
! Step (1): Find the base DN for the domain of the authenticated sender
! (username):
!
tcp_local|*|*|*|*$**@*|*   $C|BDN|$}$5,_base_dn_{|$3|$4@$5
!
! Step (2): If the base DN was found, then the probe is now
! |BDN|<username-domain-basedDN>|<auth-param>|<username>
! The following entry checks for a user entry whose canonical address or some
! alias is the authenticated sender address, with a mailRemoteIdentity
! matching the AUTH= parameter.
!
|BDN|*|*|* \
$C|LYES|$1|$}$ldap:///0?mail?sub?(&(mail=$2)(mailAlternateAddress=$2)(mailRemoteIdentity=$1))
!
! If the <username> user indeed has a mailRemoteIdentity value of the
! MAIL FROM AUTH= parameter, then the probe is now
! |LYES|<auth-param>|<username>
! If not, then the probe is still
! |BDN|<username-domain-basedDN>|<auth-param>|<username>
!
! For the "not" (still |BDN|...) case, send the message "normally"
!
|BDN|*   $Y
!
! Step (3): For the |LYES|... case, now look up the <auth-param> in the external
! LDAP directory, assumed to have a structure of external user identities
! stored under their respective domains, with attributes in the external
! user entries including:
! mail: <remote-user-address-as-in-AUTH-param>
! username: <remote-username>
! password: <remote-password>
! submittor: <local-username>
!
! This entry is checking under the domain of the <auth-param> for an entry
! with mail=<auth-param> and submittor=<username> (<username> being the local
! username), and if there is such a match returning the username and password
! attribute values.
!
|LYES|*@*|* \
$C|RYES|$0@$1|$}$extldap:///dc=$1?username?one?(&(mail=$0@$1)(submittor=$2))[\
]$extldap:///dc=$1?password?one?(&(mail=$0@$1)(submittor=$2))
!
! Step (4): If the above succeeded, the probe is now:
! |RYES|<auth-param>|<remote-username>|<remote-password>
! so now connect to the submit port (587) of a server for the <auth-param>
! domain, using smtps:, overriding the original envelope From to instead use
! the <auth-param> value, and supplying the credentials (remote username
! and remote password) that were found with the extldap: lookups.
!
|RYES|*@*|*|*   $F$Q$D$P$S|$0@$1||$2|$3|$1|587
!
! If the extldap: lookups didn't succeed, so the probe is still |LYES|... ,
! send the message "normally" (original From, etc.):
!
|LYES|*   $Y
```

```

X-FILE-IS-OLD
! The X-FILE-IS-OLD mapping table expects a probe of the form:
!   <filename>|<seconds-in-queue>
! If the filename begins with other than ZZ..., ZY..., or ZX..., or if
! the <seconds-in-queue> is greater than 3 hours, then the
! mapping returns $Y; otherwise the mapping returns $N.
! When used in a callout from another mapping table, this means that an
! X-FILENAME-IS-OLD callout will only "succeed" if the file was "old".

%%*|* \
$`("$0"!="Z"$ ||$ !find("$1","ZYX")$ ||$ integer($3)>3*60*60)?"$Y":"$N"

```

49.6 IP_ACCESS mapping table

One form of SMTP and LMTP client connection control is provided by the `IP_ACCESS` mapping table. The `IP_ACCESS` mapping table was added for JES MS 6.3.

The `IP_ACCESS` access mapping table is consulted during SMTP/LMTP client operations just prior to attempting to open connections to a remote server. (In particular, and as compared to the [AUTH_ACCESS mapping](#), the `IP_ACCESS` mapping is consulted after MX record lookup just before the A record is used.) It thus allows a "last moment" or "last ditch" check on the IP address to which the client would otherwise be about to connect---with the mapping table being able to cause the connection attempt to be aborted or redirected. This has the potential to be useful under certain special circumstances, such as security concerns where some potential destination IP address should *never* be connected to, or where it is wished to avoid connecting to known-to-be-bogus destination IP addresses (*e.g.*, 127.0.0.1 -- see also the [loopcheck channel option](#)), or where there is a desire to attempt to "fail over" to another destination IP address (similar to a [lastresort channel option](#) effect).

The `IP_ACCESS` mapping table probe has the following default format:

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

Or if bit 0 (value 1) of the (new in Messaging Server 7.0-0.04) [use_ip_access MTA option](#) is set, then the format is:

```
source-channel | ip-current-count | ip-count | ip-current-address | hostname | retry-count
```

Here `source-channel` is the channel from which the message is being dequeued, `ip-count` is the total number of IP addresses for the remote server, `ip-current-count` is the index of the current IP address being tried, `ip-current-address` is the current IP address, `hostname` is the symbolic name of the remote server, and `retry-count` is what number delivery attempt this is (based on how many prior delivery attempts, as determined by the message filename) or `-1` if the data is not available (if the message filename does not follow the MTA's normal message filenaming conventions).

The mapping can set the flags shown in [Table of IP_ACCESS mapping flags](#). In particular, setting a `$N`, `$n`, `$F`, or `$f` flag will cause the connection attempt to be aborted.

Table 49.4 IP_ACCESS mapping flags

Flag	Description
<code>\$N</code>	Immediately reject the connection attempt, hence immediately bounce the message, generating a notification message with a "Illegal host/domain name found" reason and a "5.4.4 Illegal host/domain name found" error status in

	the notification message. Any supplied text will be logged as the reason for rejection but will not be included in the DSN.
\$F	Synonym for \$N: immediately reject the connection attempt and hence the message.
\$I	Skip the current IP without attempting to connect.
\$A	Replace the current IP address with the mapping result.

Possible applications of IP_ACCESS include:

- Detect when an apparently innocuous remote destination domain name resolves in the DNS to a known "bad" or "malicious" IP address, and abort (or redirect) connections to that remote destination.
- Detect and work around internal network or configuration problems, such as "surprise" internal host name assignments, to detect and avoid host name use that might otherwise result in messages looping or being misrouted.
- Forcibly re-route (or alternatively bounce) messages that have had multiple unsuccessful delivery attempts.
- Limit the number of MX records attempted on a per-hostname basis (as opposed to use of the TCP/IP-channel-specific option [MAX_MX_RECORDS](#)).
- Call out from IP_ACCESS to a [MeterMaid table](#) to achieve "throttling" of outbound connection attempts to some specially limited remote server.

As a concrete example, here is an example of throttling outbound connections performed by a special, dedicated-to-delivery-to-a-special-destination, channel. Note that this is not generally a useful or appropriate thing to do. Limits on a remote server are its business, and under its control. Any special configuration you attempt may become out-of-date, or counter-productive in any of several ways, with no notice to you. And usually the MTA's normal scheduling, retry strategies, and load management are effective and adaptable to wide ranges of circumstances, including even unusual remote server problems. Furthermore, attempting to "work around" limits that a remote server has intentionally imposed are likely to backfire if (or more likely when) the remote server notices you "pushing" its limit and instead decides to block all your system's submissions outright. However, should such special outbound throttling still be desired...

In this example, a "special", three limited remote destination domains are assumed to be slow1.domain.com, slow2.domain.com, and slow3.domain.com, assumed to be only able to accept two hundred messages per MX host every thirty minutes.

```
! Special rewrite rules to route slow1.domain.com, slow2.domain.com, and
! slow3.domain.com out the special new tcp_outlimit channel:
!
slow1.domain.com    $U%slow1.domain.com@tcp-outlimit-daemon
slow2.domain.com    $U%slow2.domain.com@tcp-outlimit-daemon
slow3.domain.com    $U%slow3.domain.com@tcp-outlimit-daemon

! Special channel for the throttled outbound messages.
! It is a good idea to have such a special channel for handling
! the throttled outbound messages since this channel may be more prone than
! a normal channel to getting backlogged with not-yet-delivered messages.
```

```
!
tcp_outlimit smtp mx backoff pt30 ...keywords-similar-to-tcp_local...
tcp_outlimit-daemon
```

In legacy configuration:

```
metermaid.table.outthrottle.type = throttle
metermaid.table.outthrottle.data_type = ipv4
metermaid.table.outthrottle.value_type = integer
metermaid.table.outthrottle.max_entries = 50
metermaid.table.outthrottle.quota = 200
metermaid.table.outthrottle.quota_time = 1800
```

or in Unified Configuration:

```
msconfig> show metermaid.local_table:outthrottle.*
metermaid.local_table.table_type: throttle
metermaid.local_table.data_type: ipv4
metermaid.local_table.value_type: integer
metermaid.local_table.max_entries: 50
metermaid.local_table.quota: 200
metermaid.local_table.quota_time: 1800
```

IP_ACCESS

```
! For the special destination domains slow1.domain.com, slow2.domain.com,
! slow3.domain.com:
! Check the current destination IP address against MeterMaid outthrottle table.
! If over the outthrottle table's limit, this connection attempt will be
! skipped due to $I.
!
tcp_outlimit|*|*|slow1.domain.com \
$C$[IMTA_LIB:check_metermaid.so,throttle,outthrottle,$2]$I
tcp_outlimit|*|*|slow2.domain.com \
$C$[IMTA_LIB:check_metermaid.so,throttle,outthrottle,$2]$I
tcp_outlimit|*|*|slow3.domain.com \
$C$[IMTA_LIB:check_metermaid.so,throttle,outthrottle,$2]$I
!
! Otherwise, fall through and perform the connection attempt as usual.
```

49.7 TLS_ACCESS mapping table

(New in 8.0.1.) The TLS_ACCESS mapping table, if it exists, will be consulted by the [SMTP server](#) after a successful [STARTTLS negotiation](#), and by the [SMTP/LMTP client](#) after a successful [STARTTLS negotiation](#), to determine whether the MTA is happy with the STARTTLS negotiation. This allows the MTA to, for instance, decline to permit TLS use based upon a remote side's certificate issuer. If the mapping returns a N or F, then the TLS negotiation will be considered to have failed.

The probe has the form:

```
transport-info|appl-info|channel|cert-subject|cert-issuer|cert-user
```

The `channel` field will be the source channel in the case of the SMTP server, or the operating channel in the case of the SMTP/LMTP client. The `cert-user` field will be empty in the case of the SMTP/LMTP client.

Table 49.5 TLS_ACCESS mapping flags

Flag	Description
<code>\$N</code>	Force TLS negotiation failure
Flags with arguments, in argument reading order+	
<code>\$Fstring</code>	Force TLS negotiation failure, with error string <code>string</code>
<code>\$<string</code>	Send <code>string</code> to <code>syslog</code> (UNIX) if probe matches
<code>\$>string</code>	Send <code>string</code> to <code>syslog</code> (UNIX) if access is rejected

+To use multiple flags with arguments, separate the arguments with the vertical bar character, `|`, placing the arguments in the order listed in this table.

49.8 Routing via gateway systems

A local TCP/IP network may include one or more systems that are equipped to relay messages to machines not directly accessible on the local network. Such gateway systems accept addresses that are not palatable to the network itself.

One solution to this problem is to use appropriate MX records and a name resolver. However, this approach may be infeasible in some environments, so a different solution may be needed.

There is an alternate approach, in which routing to TCP/IP gateways is done by creating additional channels, one per gateway system or gateway "name" (a single gateway "name" may reference a set of gateway hosts via use of MX records), in the configuration. The name of each such channel must always begin with `tcp_`.

In Unified configuration, such a channel definition has the general form:

```
msconfig> show tcp_gateway.*
role.channel:tcp_gateway.official_host_name = gateway-system-name
role.channel:tcp_gateway.daemon = router
role.channel:tcp_gateway.smtp (novalue)
```

or equivalently

```
msconfig> show tcp_gateway.*
role.channel:tcp_gateway.official_host_name = arbitrary-placeholder-name
role.channel:tcp_gateway.daemon = gateway-system-name
role.channel:tcp_gateway.smtp (novalue)
```

In legacy configuration, the channel block for a gateway TCP/IP channel has the general form:

```
tcp_gateway smtp daemon router
gateway-system-name
```

or equivalently,

```
tcp_gateway smtp daemon gateway-system-name
arbitrary-placeholder-name
```

Rewrite rules must then be added to the configuration file to route the appropriate addresses to the gateway. See, for instance, [Routing non-local mail to a mailhub](#).

The "daemon router" setting tells the SMTP client program not to open a connection directly to the first system named in the envelope address list, but to instead open a connection to the official host for this channel, `gateway-system-name`. Usually the default `multiple` setting is appropriate and desirable on gateway channels; but certain gateways may restrict the number of addresses that can appear in a single copy of a message, in which case it may be appropriate to add either the `single` or `single_sys` setting to those gateway channels. If the gateway can handle multiple simultaneous connections, then use of the `threaddepth` setting may be of interest to cause outgoing connections to be split amongst multiple threads.

Once a channel block for a gateway is created the channel should be ready to use.

Note that when addresses are being looked up in LDAP in a so-called "direct LDAP" [configuration](#), then certain domain-level LDAP attributes including `mailRoutingHosts` and `mailRoutingSmartHost` (or more precisely, the LDAP attributes named by the MTA options `ldap_domain_attr_routing_hosts` and `ldap_domain_attr_smarthost`), may also potentially be used for similar route-to-gateway-host purposes. And so-called "detour host" functionality, typically used for purposes of routing through an SMTP host performing spam/virus filtering, also has a routing effect.

Typically, use of a `daemon` channel is especially appropriate for routing outbound (to the Internet) messages, when all such messages should go out through a gateway. In contrast, use of LDAP-based domain routing attributes is especially appropriate for controlling the routing to internally-destined messages, when such messages should go through an internal "smart host" that performs additional address handling. And use of "detour host" functionality is, as already mentioned above, intended for cases of detour routing through spam/virus filter boxes.

49.8.1 Routing non-local mail to a mailhub

Sometimes it is convenient to configure the MTA to route mail not for the local host, or a group of local machines, to a central machine and leave it up to that machine to deal with the mail, perhaps relaying it to the outside world or other local machines, or perhaps even gatewaying it into other mail systems. The following example configuration, [Routing messages to a central system](#), illustrates doing just this.

In this example, the local host is `host1.domain.com` and two other local machines, `host2.domain.com` and `host3.domain.com`, are recognized. Mail for either of those two machines is sent via a `tcp_local` channel (SMTP over TCP/IP) to those hosts. All other mail not for `host1`, `host2`, or `host3` is sent via another SMTP over TCP/IP channel, named `tcp_gateway`, to the host `mailhub.domain.com`. A "match-all" rewrite rule is used to direct all mail not for `host1`, `host2`, or `host3` to that channel. (The match-all rewrite rule is described in [A rule to match any address](#).) The `daemon` channel option is used with the `tcp_gateway` channel, telling the channel to routed messages queued to it through the host `mailhub.domain.com`. For additional discussion of such usage, see also [Routing via gateway systems](#).

A legacy configuration example of routing to a central system would be:

```

!
! Rewrite rules for the local host/cluster
!
host1                $U@host1.domain.com
host1.domain.com     $U@host1.domain.com
!
! Rewrite rules for some internal systems
!
host2.domain.com     $U%host2.domain.com@TCP-DAEMON
host3.domain.com     $U%host3.domain.com@TCP-DAEMON
!
! Use a match all rewrite rule to route everything
! else to the mailhub.domain.com
!
.                    $U%$H@mailhub.domain.com$A

l
host1.domain.com

tcp_local smtp single_sys mx
TCP-DAEMON

tcp_gateway smtp mx daemon router
mailhub.domain.com

```

49.9 Blocking SMTP relaying

One application of the [ORIG_SEND_ACCESS mapping table](#), along with appropriate configuration of channels with [switchchannel](#) and [saswitchchannel](#) channel options, is to prevent people from using your MTA to relay junk mail to hundreds or thousands of Internet mail boxes. By default, at a code level, the MTA does not prevent SMTP relaying activity. However, the normal installation does configure to prevent SMTP relaying from "external" sources.

Blocking unauthorized relaying while allowing it for legitimate local users requires configuring the MTA to know how to distinguish between the two classes of users. The distinction between "internal" *vs.* "external" users is achieved using a combination of the [switchchannel](#) and [allowswitchchannel](#) channel options in conjunction with a rewrite rule that compares connection source IP addresses against the [INTERNAL_IP mapping table](#), plus user authentication during message submission in conjunction with the [maysasl](#) and [saswitchchannel](#) channel options. The effect of such configuration is to "sort" incoming messages based on knowledge of the message's source (whether that knowledge is of the source of the IP connection, or of the sender of the messages) assigning the messages to different source channels. Once such assignment of source channel has been achieved, then the desired relaying restrictions can be easily controlled in, *e.g.*, the [ORIG_SEND_ACCESS](#) mapping table.

When preventing unauthorized people from relaying SMTP mail through your system, keep in mind that you *do* want to allow local users to relay SMTP mail! For instance, POP and IMAP users typically rely upon using the MTA 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. It's those random people out on the Internet who you want to prevent from using you as a relay. With appropriate configuration to recognize the cases of "internal" source IP addresses or local user authenticated messages and handle them via special channels, rather than the default `tcp_local` channel, you can differentiate between these classes of users and block only the correct class. Specifically, (once "internal" users have been properly sorted for handling by one or another special incoming channel), 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 may be conveniently 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 the above, 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.

Note that 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 l (lowercase "L") 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.

A further refinement, to block attempts to relay "through" internal systems using source-routed addresses, is included in the following sample mapping table:

```
ORIG_SEND_ACCESS
```

```
tcp_local|*|tcp_local|*                $NRelaying$ not$ permitted
!
! Block direct submission to MTA "intermediate" channels
!
tcp_*|*|native|*                        $N
tcp_*|*|hold|*                          $N
tcp_*|*|pipe|*                          $N
!
! Block direct submission to Message Store delivery channels;
! routing to such channel should only occur due to MTA address/alias
! processing
!
tcp_*|*|ims-ms|*                        $N
tcp_*|tcp_lmtpcs*|*                    $N
!
! Block "external" submissions of explicitly source-routed "internal" addresses
!
tcp_local|*|tcp_intranet|@*:*.*        $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|*%*.*        $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|*.*!*.*      $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|"*@*.*        $N$D30|Explicit$ routing$ not$ allowed
```

The final four entries will cause attempts by remote senders to submit explicitly source-routed messages to be immediately rejected. Note that depending upon the actual configuration of your other, internal hosts, it is often the case that even such explicitly source-routed attempts at relaying will not, in fact, end up truly getting relayed back out to the Internet, but will instead be rejected by the other, internal host. But by rejecting the attempts immediately, you can avoid getting (falsely) accused of being an open relay by carelessly constructed and operated relay testers, that only check if their relay attempt was initially blocked, rather than testing whether the probe message actually ever got relayed out and delivered.

49.9.1 SRS and Relay Blocking

Prior to the 8.0 release, decoding of SRS addresses happened invisibly before all other address processing (including probing of access mapping tables such as `ORIG_SEND_ACCESS`), with the result that when a remote site bounced a message from an SRS encoded sender address, the notification message returning to the encoded SRS address came to the MTA which decoded the address to (typically) discover a remote sender address and potentially reject the notification message as an attempt to "relay" (a notification message from a remote site to a remote original sender, in its attempt to pass through the MTA). As of 8.0, the still-SRS-encoded address is used in the `ORIG_SEND_ACCESS` probe, nullifying this problem. Meantime, in earlier versions, there is an approach to work around this problem.

Configuring with the `access_orcpt=2` and modifying the entries of the `ORIG_SEND_ACCESS` mapping table to expect an ORCPT field in each probe is one way to work around such an issue. In the following example, all the "usual" entries of `ORIG_SEND_ACCESS` have been modified to expect an additional field in the probe, the "orcpt" field, and an initial entry (prior to the basic `tcp_local -> tcp_local` block entry) has been added to allow passing through addresses that turn out to be "remote" when the MTA's own SRS encoding is removed:

```
ORIG_SEND_ACCESS

! Allow "relaying" of responses (such as notification messages) back to
! those original messages that came from remote senders to originally local
! recipients which the MTA relayed onwards, SRS-encoded, to remote recipients.
! That is, these are messages (notification messages) from remote sites to
! which local users had forwarded their e-mail, back to original senders to
! those (forwarding) local users:
! such messages that come in addressed using an SRS encoding with this MTA's
! own srs_domain, but which (once SRS encoding is removed) end up addressed
! back to a "remote" address.
!
tcp_local|*|tcp_local|*|rfc822;SRS0=*<srs_domain>      $Y
!
! Normal relay blocking entry
!
tcp_local|*|tcp_local|*|*                            $NRelaying$ not$ permitted
!
! Block direct submission to MTA "intermediate" channels
!
tcp_*|*|native|*|*                                   $N
tcp_*|*|hold|*|*                                     $N
tcp_*|*|pipe|*|*                                     $N
!
```

```
! Block direct submission to Message Store delivery channels;
! routing to such channel should only occur due to MTA address/alias
! processing
!
tcp_*|*|ims-ms|*|*           $N
tcp_*|tcp_lmtpcs*|*|*       $N
!
! Block "external" submissions of explicitly source-routed "internal" addresses
!
tcp_local|*|tcp_intranet|@*:*.*|*   $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|*$%*@*|*   $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|*.*!*@*|*  $N$D30|Explicit$ routing$ not$ allowed
tcp_local|*|tcp_intranet|"*@*"*@*|* $N$D30|Explicit$ routing$ not$ allowed
```

49.10 Triggering message transfer with remote SMTP systems

In cases where the network connection between two systems is only available at particular times--a "dial up" sort of connection for instance--there is an SMTP extension whereby one system can inform another that it is ready to receive mail. This is performed using the SMTP extension command ETRN, defined in [RFC 1985](#): the side that desires to receive mail connects to the remote side's SMTP server and issues the command ETRN *receivinghostname*. If the remote side's SMTP server supports the ETRN command, it will then attempt delivery of any messages it has waiting to be sent to *receivinghostname*.

The MTA's SMTP server supports ETRN. In particular, the SMTP server interprets a received ETRN *hostname* command as a request to run the channel which *hostname* matches, a received ETRN *@hostname* as a request to deliver all messages in the *hostname* subnet, and a ETRN *#channelname* command as a request to run the channel *channelname*. By default, the SMTP server always responds to remote side's ETRN requests; if you wish to restrict this behavior, see [disableetrn](#) and related channel options, or the [ETRNL_ACCESS mapping table](#).

And outgoing SMTP-based channels, such as TCP/IP channels, can be configured to send an ETRN command at the beginning of an outgoing SMTP dialogue via the [sendetrn](#) channel option. For instance, suppose a system *host1.acme.com* has a dial-up connection to a remote system *intermittent.some.where.com*, where the *intermittent.some.where.com* system also supports ETRN. For a channel for connecting up to the remote side and sending ETRN, such a site might use a channel definition along the lines of:

```
tcp_dialup smtp mx daemon intermittent.some.where.com \
  periodic sendetrn host1.domain.com
TCP-DIALUP
```

As of iMS V5.0, there is an ETRNL_ACCESS mapping table. Probes have the form:

```
transport-info | app-info | channel-to-run | full-name | claimed-system
```

(Here *claimed-system* is the ETRNL parameter, and *full-name* is a processed version of that parameter.) If the mapping table returns a \$N, \$n, \$F, or \$f, then the ETRNL command is rejected with a 459 4.5.0 error; by default

459 4.5.0 Cannot start delivery on channel - access denied

or if alternate text is included in the \$N entry, then the alternate text is used.

If the ETRN_ACCESS mapping table returns a \$S or \$s, then the ETRN is attempted. If the mapping table also returns a \$Dchannel-name or \$dchannel-name, then the MTA tries to lookup channel-name (in the channel/host table from the configuration file) and if that lookup is successful, runs that channel (rather than whatever channel the original ETRN command might have run).

49.11 Authentication errors and resultant SMTP errors

The authentication code performs various checks on the user account when attempting to authenticate, as for instance during SMTP AUTH processing. This may result in authentication errors being returned to the SMTP server, which will in turn issue an SMTP error back in response to the SMTP AUTH attempt. Errors of note include the following.

If the client's SMTP AUTH attempt uses either a bad username or a bad password, or the authentication mechanism is too weak for site policy, the SMTP server will issue the (same for each case) error response:

535 5.7.8 Bad username or password

though the SMTP server will optionally (`log_message_id=1` and `log_connection`'s bit 7/ value 128 set) record the real cause of the authentication failure (respectively, "No such user" or "Bad password" or "Authentication mechanism is too weak") in the message-id field of the "U" [connection transaction log](#) entry.

If the LDAP attribute `mailAllowedServiceAccess` has been set to disallow SMTP access, the authentication attempt will be rejected with:

535 5.7.8 Authorization failure

If using this feature with the goal of disallowing certain users from sending messages, note that it is critically important to first configure so that users are *required* to use SMTP AUTH when submitting (see the `mustsaslserver` channel option); otherwise, in preventing certain users from sending when they properly authenticate, the unintentional (and undesirable) effect is likely to be to discourage those users from attempting authentication, instead effectively encouraging those users to send without authentication!

If the user's LDAP attribute `mailUserStatus` is set to `inactive` or `disabled`, then the SMTP error will be:

525 5.7.13 Account disabled

with, if [MTA connection transaction logging](#) is enabled and in particular if the the optional SASL attempt logging is enabled, then in the resulting "U" connection transaction entry the message-id field will include additional detail: either "Account disabled (inactive)" or "Account disabled (hold)".

There are additional errors that may be returned, as for syntax problems in the client's SMTP AUTH command, or SASL mechanism problems, including:

501 5.7.0 Cannot decode BASE64
504 5.5.4 Unrecognized authentication type
501 5.5.0 Invalid input
523 5.7.10 Encryption needed to use mechanism
524 5.7.11 Password expired, has to be reset

Temporary LDAP errors will result in a temporary SMTP error:

454 4.7.0 Authentication server unavailable



Chapter 50 BSMTP channels

50.1 Configuring the BSMTP channels	50-1
50.2 BSMTP service conversions	50-4

Batch SMTP (BSMTP) is a batch-mode implementation of the SMTP protocol which turns SMTP into a remote-submission protocol. For over a decade, batch SMTP was used quite heavily as a message transfer protocol on the international BITNET network. Cooperating MTA sites can use BSMTP as an effective means of moving mail in bulk between one another; for instance the exchange of company e-mail between two company offices by means of the Internet.

That is, it is possible to tunnel messages between two or more cooperating MTA systems using Batch SMTP (BSMTP). In addition, the MTA's general service conversion facilities can be used to provide services such as payload compression and digital signatures for authentication and integrity.

With BSMTP, messages are bundled together on one MTA system and then periodically transmitted through arbitrary MTAs and networks to a remote MTA system. Upon receipt at the remote system, the bundle is unpacked and the individual messages sent on to their recipients. Note that the bundled (encapsulated) handling of messages with BSMTP has the following inherent aspects which may be useful in some contexts:

1. The original message envelope information (sender, recipient, *etc.*) is not visible at the SMTP level on the intermediate MTAs; instead, the generic BSMTP addressing information is all that is visible at the outer message level.
2. The original message headers are transferred unaltered from the point of initial sending system's BSMTP bundling (encapsulation) until the destination system's BSMTP unbundling (message extraction) is performed.

Such aspects may be of interest in cases where it is desired to make *traffic analysis* (analysis of who is sending how much e-mail to whom) difficult, or when it is desired to protect original message headers from intermediate MTA processing, or desired to avoid exposing the intermediate MTA hops (*e.g.*, additional Received: header lines) to the final recipients. Furthermore, with the MTA's general service conversion facilities, arbitrary transformations can be performed on the bundles such as document conversion, compression, addition of digital signatures for authentication and integrity, *etc.*

50.1 Configuring the BSMTP channels

Each of the MTA systems which will be exchanging mail via BSMTP will need one incoming BSMTP channel and an outgoing BSMTP channel for each of the remote MTA systems. The channel definitions should be along the lines of:

```
bsin_gateway smtp  
bsin.host0
```

```
bsout_remote1 smtp master user bsmtpl daemon host1  
BSOUT-REMOTE1
```

```
bsout_remote2 smtp master user bsmtplib daemon host2
BSOUT-REMOTE2...
```

```
bsout_remoteN smtp master user bsmtplib daemon hostN
BSOUT-REMOTEN
```

where *host0* is the name of the local MTA host, as used by the other remote MTA systems, and *host1*, *host2*, ..., *hostN* are the host names of the remote MTA systems. The strings *remotel*, *remote2*, ... *remoten* and *REMOTE1*, *REMOTE2*, ..., *REMOTEN* are arbitrary and need just be distinct from one another.

With the above definitions, the channel *bsout_remotel* will bundle up its BSMTP parcels and send them on to the fixed address *bsmtplib@host1*. Likewise for the remaining BSOUT channels.

The rewrite rules appear as:

```
domain1      $U%H@BSOUT-REMOTE1$Nbsout_remotel
.domain1     $U%H$D@BSOUT-REMOTE1$Nbsout_remotel
domain2      $U%H@BSOUT-REMOTE2$Nbsout_remote2
.domain2     $U%H$D@BSOUT-REMOTE2$Nbsout_remote2
...
domainN      $U%H@BSOUT-REMOTEN$Nbsout_remoten
.domainN     $U%H$D@BSOUT-REMOTEN$Nbsout_remoten
```

where *domain1*, *domain2*, ... *domainN* are the domain names of the remote MTA systems.

Finally, add to the [FORWARD mapping table](#) the entry

```
FORWARD
    bsmtplib@host0    bsmtplib@bsin.host0$Y$D
```

where, again, *host0* is the host name for the local MTA system which will be used by the BSOUT channels on the remote MTA systems. That way, when they send BSMTP parcels to *bsmtplib@host0*, it will be forwarded on to the local *bsin_gateway* channel.¹

For example, assume that the *domain.com* domain will be exchanging BSMTP traffic with the *domain.co.uk* domain via the MTA hosts *hub.domain.com* and *athena.domain.co.uk*. Then *hub.domain.com* would have the configuration

```
domain.co.uk    $U%H@BSOUT-REMOTE1$Nbsout_remotel
.domain.co.uk   $U%H$D@BSOUT-REMOTE1$Nbsout_remotel
```

...

```
bsin_gateway smtp
bsin.hub.domain.com
```

```
bsout_remotel smtp master user bsmtplib daemon athena.domain.co.uk
BSOUT-REMOTE1
```

and the [FORWARD mapping table](#) entry

```
FORWARD
```

```
bsmtp@hub.domain.com bsmtp@bsin.hub.domain.com$Y$D
```

The system athena.domain.co.uk would have the configuration

```
domain.com      $U%$H@BSOUT-REMOTE1$Nbsout_remotel
.domain.com     $U%$H$D@BSOUT-REMOTE1$Nbsout_remotel
```

```
...
```

```
bsin_gateway smtp
bsin.athena.domain.co.uk
```

```
bsout_remotel smtp master user bsmtp daemon hub.domain.com
BSOUT-REMOTE1
```

and the [FORWARD mapping table](#) entry

```
FORWARD
```

```
bsmtp@athena.domain.co.uk bsmtp@bsin.athena.domain.co.uk$Y$D
```

With the above configurations, when a user on hub.domain.com sends mail to user@domain.co.uk, the message is routed to the `bsout_remotel` channel. That channel will package the message up into a BSMTP parcel and send that parcel on to bsmtp@athena.domain.co.uk. Owing to the `$Nbsout_remotel` tag in the domain.co.uk rewrite rules, those rewrite rules will be ignored when the `bsout_remotel` channel enqueues the message. Instead, the normal rewrite rules for domain.co.uk will take effect and route the message containing the parcel out to the WAN (*e.g.*, the Internet).

Note that the outbound BSMTP channels can construct application/batch-smtp message parts containing multiple messages. As such, sites may wish to use the [after channel option](#) on their BSOUT channels. So doing may prove advantageous for sites who wish to bundle their mail up into large parcels and send those parcels only once every few minutes, hours, or days. Also, the [ATTEMPT_TRANSACTIONS_PER_SESSION](#) TCP/IP-channel-specific option might be used with the BSOUT channels to prevent cases where, under heavy load, a BSOUT channel just runs continuously bundling into a single parcel messages queuing up to be sent out. This option puts an upper limit on the number of messages placed in a single parcel and forces the channel to close a parcel, send it along, and start a new parcel when there are lots of messages to bundle up.

This completes the basic configuration so that BSMTP channels may run and deliver messages. Commonly, however, sites also desire to perform one or more forms of message transformation or processing on BSMTP messages; for further details, see [BSMTP service conversions](#).

Note ¹ Any of several mechanisms might be used to accomplish this forwarding. The most efficient is the use of an alias when `host0` is the official local host name for the MTA system. The least efficient is the [FORWARD mapping table](#); which method is best for a given site depends upon site-specific issues. Use of the FORWARD mapping table is presented here because that method works in all cases.

50.2 BSMTP service conversions

The MTA's [service conversion](#) facility may be used with BSMTP channels to perform desired message transformations on incoming and outgoing messages.

Usually outgoing BSMTP channels, BSOUT channels, are configured to perform one sort of service conversion on the messages they emit, and incoming BSMTP channels, BSIN channels, are configured to perform the inverse service conversion on messages they receive. Thus when BSMTP channels are used, the configuration would also usually contain a [CHARSET-CONVERSION mapping](#) such as:

```
CHARSET-CONVERSION
```

```
in-chan=bsout_*;out-chan=*;convert      yes
in-chan=*;out-chan=bsin_*;convert      yes
```

whether in the MTA mappings file in legacy configuration, or in Unified Configuration alternatively appearing as:

```
msconfig> show mapping:CHARSET-CONVERSION
role.mapping:CHARSET-CONVERSION.rule = in-chan=bsout_*;out-chan=*;convert yes
role.mapping:CHARSET-CONVERSION.rule = in-chan=*;out-chan=bsin_*;convert yes
```

Note that the `CHARSET-CONVERSION` entries shown are such as to enable service conversions for messages sent from BSOUT channels (such as messages transitting through a BSOUT channel on their way out to an outgoing TCP/IP channel), as well as for messages sent to a BSIN channel (such as messages transitting through a BSIN channel on their way in from an incoming TCP/IP channel).

Once execution of service conversions has been enabled via a `CHARSET-CONVERSION` mapping such as that shown above, the specific service conversions to be performed must be configured: whether as conversions entries in Unified Configuration, or in the MTA conversions file in legacy configuration. Section 19.2.3 provides examples of configuring specific service conversions on UNIX.

Chapter 51 ims-ms channels

51.1	ims-ms channel configuration	51-1
51.1.1	Additional ims-ms channels	51-4
51.2	ims-ms-channel-specific options	51-5
51.2.1	DEBUG ims-ms-channel-specific option	51-6
51.2.2	DELIVER_THREADS ims-ms-channel-specific option	51-6
51.2.3	FILEINTO ims-ms-channel-specific option	51-6
51.2.4	LIFETIME_CAPACITY ims-ms-channel-specific option	51-7
51.2.5	LOG_DEQUEUE_RATE ims-ms-channel-specific option	51-7
51.3	ims-ms channel program switches	51-7
51.4	ims-ms channel debugging	51-7

ims-ms channels deliver messages to the Messaging Server Message Store. Normally, only one ims-ms channel is needed. But for special purposes, it is possible to define and use [additional ims-ms_* channels](#).

51.1 ims-ms channel configuration

An appropriate initial ims-ms channel definition and rewrite rule(s) are normally set up by the initial Messaging Server MTA installation and configuration process. Such a channel definition usually looks something like:

```
msconfig> show channel:ims-ms
role.channel:ims-ms.official_host_name = ims-ms-daemon
role.channel:ims-ms.backoff = PT5M PT10M PT30M PT1H PT2H PT4H
role.channel:ims-ms.defragment (novalue)

role.channel:ims-ms.fileinto = $U+$S@$D
role.channel:ims-ms.maxjobs = 2
role.channel:ims-ms.notices = 1 7 14 21 28
role.channel:ims-ms.pool = IMS_POOL

role.channel:ims-ms.subdirs = 20
```

or in legacy configuration:

```
ims-ms defragment subdirs 20 notices 1 7 14 21 28 \
backoff "pt5m" "pt10m" "pt30m" "pt1h" "pt2h" "pt4h" \
maxjobs 2 pool IMS_POOL fileinto $U+$S@$D
ims-ms-daemon
```

Because the ims-ms channel is, on a typical system hosting a Message Store, such an important channel in the overall picture of message handling, here is a brief synopsis of these typical ims-ms [channel option](#) usages, discussed in turn.

- ims-ms channels should all normally be marked with the [defragment](#) channel option so that any incoming MIME fragmented messages will take a detour through the [defragment channel](#) to get an attempt at MIME message reassembly before being delivered to the Message Store.

- The `ims-ms` channel is typically a heavily used channel, at least on a system that hosts a Message Store with numerous active users. For a heavily used such channel, it is typically a good idea to configure with a generous number of subdirectories for performance reasons; thus the use of the `subdirs` option. On an especially busy Message Store system, where `user quotas are enforced` but a generous `quota grace period` is allowed, an even higher value for `subdirs` may be desirable. On a system that is only, or primarily, an SMTP relay host, that does not have a significantly used message store, use of `subdirs` may be unnecessary and the option might be removed.
- With the MTA option `return_units` at its (default) setting of 0 so that `notices` values are interpreted in units of days, the `ims-ms` channel is often set to retain messages for a rather long period, intended to cover the overquota grace period--that is, give users a chance to clean out old messages and receive their additional messages -- before eventually returning (bouncing) the messages as undeliverable. That is, a special setting rather more generous than the setting used for other channels (such as channels attempting to deliver out to the Internet) is often used.
- Fairly "rapid" initial additional delivery attempts are often configured for the `ims-ms` channel, via small initial `backoff` values. Since a user mailbox may get temporarily locked by the Message Store while another message is being delivered, or while a user is moving or copying a message to another folder via IMAP, it is possible to see short-term inability to deliver to a particular mailbox that will resolve relatively quickly. Thus it is worthwhile to re-attempt delivery at fairly rapid initial intervals. (This is unlike the case of a channel attempting to deliver to remote Internet hosts, where remote hosts may be unreachable for hours or days and hence where very rapid repeated delivery attempts may be useless and often even counter-productive, and where furthermore, Internet standards require waiting at least one half hour before reattempting message delivery to remote Internet hosts.)
- Due to the importance, in a typical deployment, of the `ims-ms` channel, a separate `Job Controller pool` intended for the sole use of the `ims-ms` channel is usually defined, and then the `ims-ms` channel is configured to run in that processing pool via the `pool` keyword. In particular, in this way the `ims-ms` channel runs in a separate Job Controller processing pool than `TCP/IP channels` (another heavily used type of channel), or than internal processing channels such as the `process channel` (used for processing `notification messages`, hence subject to periods of heavy use); the potential for competition for resources between these different types of channels is limited thereby. The `maxjobs` channel option is used to limit the number of simultaneously running `ims-ms` channel processes. Note that the `ims-ms` channel is multi-threaded; see the `DELIVER_THREADS` `ims-ms-channel-specific option`. So even a single `ims-ms` channel process can be attempting multiple deliveries at once. The Job Controller attempts to sort messages destined for a particular user into the same processing thread; this limits mailbox locking contention between different processing threads. (For another factor that can affect `ims-ms` channel performance, see the `threaddepth` channel option.)
- The `fileinto` channel option is used to specify support for Sieve `fileinto` actions. (That is, the `fileinto` channel option is used to specify what a Sieve `fileinto` action actually causes to occur--namely, changing the address to include the folder name as a subaddress. The `ims-ms` channel by default then interprets the subaddress as a request to deliver to the user's named folder, unless disabled via the `ims-ms-channel-specific option FILEINTO`.)

Rewrite rules for the `ims-ms` channel involve more complexity than the rewrite rules for many other types of channels, as they are fundamentally intertwined with direct LDAP lookup processing. This discussion will not attempt to cover the whole of direct LDAP

lookup processing; instead, this discussion will merely provide a brief overview (simplifying and omitting some details) of the basics of the rewriting involved in routing to the `ims-ms` channel.

Basic rewrite rules relevant for the `ims-ms` channel are normally set up by the initial Messaging Server MTA installation and configuration process. Those rewrite rules usually look something like:

```
! Basic direct LDAP rewrite rule to select local users
!
$* $A$E$F$U%$H$V$H@local-channel-official-host-name
!
! ...various other rewrite rules...
!
! ims-ms
.ims-ms-daemon $U%H.ims-ms-daemon@ims-ms-daemon
```

where `local-channel-official-host-name` is the official host name (first name on the second logical line of the channel definition) on the local ("l", lowercase ell) channel definition.

In Unified Configuration, this would appear as:

```
msconfig> show rewrite * $*
role.rewrite.rule = $* $A$E$F$U%$H$V$H@&/IMTA_HOST/
msconfig> show rewrite * .ims-ms-daemon
role.rewrite.rule = .ims-ms-daemon $U%H.ims-ms-daemon@ims-ms-daemon
```

where note that the `&/IMTA_HOST/` handles the insertion of the `ldap_local_host` value (which normally should match the `channel:1.official_host_name` value).

Also extremely relevant for the rewriting process for the `ims-ms` channel, and normal routing of messages to the `ims-ms` channel, are the MTA options `alias_urlN`, especially `alias_url0`, and `delivery_options`, especially its `mailbox` clause, normally set to:

```
*mailbox=$M%$\'$2I$_+$2S@ims-ms-daemon
```

(And security-related settings are the use of the `viaaliasrequired` channel option on the local channel, and the `ORIG_MAIL_ACCESS mapping table` entry that blocks direct submission to `username@ims-ms-daemon` sorts of addresses; that is, these two configuration choices add in restrictions to mean that "local" addresses must have an alias in LDAP, and route to the `ims-ms` channel by way of an alias expansion.)

The normal process of routing messages to the `ims-ms` channel involves rewriting, alias expansion, and application of user LDAP attributes, as follows:

1. Initially, the domain in an envelope To address (recipient address) is used to do an LDAP search for the domain; this is done via the (direct LDAP domain lookup) rewrite rule:

```
$* $A$E$F$U%$H$V$H@local-channel-official-host-name
```

2. If the domain was found in LDAP, various domain level attributes are set and this rewrite rule (when the LDAP lookup succeeds) forcibly matches the address to the local (lowercase l) channel.
3. When an address matches the local (lowercase "l") channel, the MTA performs alias expansion on the address. In particular, this includes an LDAP lookup (the [alias_url0](#) lookup, and if other `alias_urlN` lookups are configured, then if necessary those lookups also) attempting to find an LDAP entry for this user address.
4. When a user entry is found in LDAP, the values of its `mailDeliveryOption` LDAP attribute (more specifically, the LDAP attribute named by the MTA option [ldap_delivery_option](#)) are inspected. A value of `mailbox` will cause the configured rule for the `mailbox` clause of the [delivery_options](#) MTA option to be applied; namely, the address will be converted to the form (if no subaddress/folder name is present):

```
uid%lowercased-domain-name@ims-ms-daemon
```

or if a subaddress/folder name is present:

```
uid%lowercased-domain-name+folder@ims-ms-daemon
```

Or if the domain name is actually the default domain name (as specified via either the [defaultdomain option](#) -- in legacy configuration, the `configutil` parameter `service.defaultdomain`, or the MTA option [ldap_default_domain](#)), then the domain name (and leading percent character) are omitted, hence:

```
uid@ims-ms-daemon
```

or

```
uid+folder@ims-ms-daemon
```

5. This (transformed) address then goes through rewriting, and it forcibly matches the `ims-ms` channel, due to the matching official host name (`ims-ms-daemon`) for the channel: the message is routed to the `ims-ms` channel.

51.1.1 Additional ims-ms channels

It is possible to define additional `ims-ms_*` channels, if special needs would make additional such channels useful. Each such channel should have a name of the form `ims-ms_distinguishing-string`, and its own unique official channel host name. Note that due to the intimate connection between rewriting, alias expansion, and delivery options for `ims-ms` channels, getting additional `ims-ms_*` channels used by the MTA typically requires modification to other configuration choices also. For instance, one approach would be as follows.

In this example a new channel `ims-ms_shortterm` will have a [shorter retention policy](#) for holding onto messages that could not be delivered, while a new channel `ims-ms_vip` will have a [faster delivery retry rate](#), and will run in its own [Job Controller processing pool](#) so as not to compete with the "regular" `ims-ms` channel and the `ims-ms_shortterm` channel.

First define additional `ims-ms_*` channels such as:

```

ims-ms_shortterm defragment subdirs 20 notices 1 7 14 \
  backoff "pt5m" "pt10m" "pt30m" "pt1h" "pt2h" "pt4h" \
  maxjobs 2 pool IMS_POOL fileinto $U+$S@$D
ims-ms-short-daemon

ims-ms_vip defragment subdirs 20 notices 1 7 14 21 28 \
  backoff "pt5m" "pt5m" "pt10m" "pt10m" "pt30m" "pt1h" "pt2h" \
  maxjobs 2 pool IMS_VIP_POOL fileinto $U+$S@$D
ims-ms-vip-daemon

```

Next tell the MTA about two new delivery option values, `mailboxshort` and `vipmailbox`, defining them in the MTA option `delivery_options`:

```

DELIVERY_OPTIONS=*mailbox=$M%\$2I$_+$2S@ims-ms-daemon,\
&members=*,\
 *mailboxshort=$M%\$2I$_+$2S@ims-ms-short-daemon,\
 *vipmailbox=$M%\$2I$_+$2S@ims-ms-vip-daemon,\
 *native=$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

```

(In the above setting, note the use of a leading space before each continued line of the option value to avoid causing interpretation of characters such as `#` as a comment character. Note also that the site-specific definitions for `mailboxshort` and `vipmailbox` are added after the usual first two definitions for `mailbox` and `members` respectively, as these first two value clauses in the `delivery_options` definition have special meaning as far as being defaults.)

Finally, set appropriate users' LDAP entries with a value for the `mailDeliveryOption` attribute (or more precisely whatever attribute is named by the `ldap_delivery_option` MTA option) of `mailboxshort` or `vipmailbox` as desired.

51.2 ims-ms-channel-specific options

`ims-ms channels` support, in addition to the usual `channel options`, a number of `ims-ms-channel-specific options`. These `ims-ms-channel-specific options` are set in legacy configuration in a channel-specific option file, or in Unified Configuration are set under the channel's `options` option. For instance:

```
msconfig> set channel:ims-ms.options.DEBUG 4
```

Note that in Unified Configuration such channel-specific options (those set under the `options` option) are not (currently) schema checked: be careful when setting them as `msconfig` will not warn of invalid values or syntax as it would for regular channel options! (Nor will `msconfig` show whether or not such channel-specific options have any default value.)

In legacy configuration, where an option file is used, such an option file must be named `x_option` where `x` is the name of the channel, and stored in the MTA table directory. Hence the name of the `ims-ms` channel option file is `ims-ms_option`, *i.e.* `msg-root/config/ims-ms_option`.

51.2.1 ims-ms-channel-specific options: DEBUG (integer)

The `DEBUG` `ims-ms-channel-specific` option can take values between 0, the default, up to 4; increasing values mean increasing amounts of debug output being written to the `imta` log file. The default of 0 means no debugging information. A value of 3 or more causes inclusion of information about the numbers of messages waiting and numbers of threads in use, and whether more threads must be started, and will also cause output when a thread exits, and if a shutdown request has been received. A value of 4 or more causes debug output regarding which message is currently being processed (specifically, the envelope `From` address for the current message), and debugging that such a message has been successfully delivered.

The debug output from setting `DEBUG` is normally directed to the `imta` file (not the `ims-ms_master.log-*` file), but will only be actually written to `imta` if the `loglevel` option for the MTA (in legacy configuration, the `logfile.imta.loglevel` configutil parameter) is set to the value `debug`. (And potentially, if the `syslogfacility` option is set `--logfile.imta.syslogfacility` configutil parameter in legacy configuration `--` then the output is instead directed to `syslog`.)

Note that setting the `master_debug` channel option on an `ims-ms` channel also forces a `DEBUG=4` level setting. `master_debug` itself will cause additional output, to a different log file, than the `DEBUG` `ims-ms-channel-specific` option; `master_debug` causes MTA processing debug output to be written to a `channel-name_master.log-*` file, normally an `ims-ms_master.log-*` file.

As of JES MS 6.2, see also the `activate` Message Trace option, which if set to `yes` will cause Message Store message tracing information to be written to the `msgtrace` file.

51.2.2 ims-ms-channel-specific options: DELIVER_THREADS (integer)

The `DELIVER_THREADS` `ims-ms-channel-specific` option controls the maximum number of delivery threads used by an individual `ims-ms` delivery process. The default is 15, unless the `command line -t switch` has been used to specify a different value. This option overrides such a `command line -t switch`.

51.2.3 ims-ms-channel-specific options: FILEINTO (0 or 1)

The `FILEINTO` `ims-ms-channel-specific` option controls whether subaddresses are interpreted as folder names for delivery purposes by the channel. The default is 1, meaning that such folder delivery is enabled. If this option is set to 0, then folder delivery is disabled.

This channel-specific option is not normally changed from the default value; but if it is, see also the general `fileinto` channel option, with whose setting this `ims-ms-channel-specific` option should be coordinated.

51.2.4 ims-ms-channel-specific options: LIFETIME_CAPACITY (integer)

The `LIFETIME_CAPACITY` ims-ms-channel-specific option specifies the maximum number of message files that a thread will handle before exiting. The default value is `-1`, meaning no limit. Note that this count of message files is based on the number of files handed to the channel, and that an individual message file, while from a single sender, may be destined for multiple recipients; in particular, the count is not based on the number of message recipients.

51.2.5 ims-ms-channel-specific options: LOG_DEQUEUE_RATE (0 or 1)

The `LOG_DEQUEUE_RATE` ims-ms-channel-specific option controls whether the channel logs the message dequeue rate. The default is `0`, unless the [command line `-d` switch](#) has been used in which case the default is `1`. The default of `0` means not to log message dequeue rate; a value of `0` means to log message dequeue rate; if the [DEBUG](#) level has been set to `2` or more, the message dequeue rate is periodically logged, rather than just a final summary being output.

51.3 ims-ms channel program switches

The [Job Controller's default configuration](#) has the Job Controller execute the `ims_master` channel program with no arguments:

```
msconfig> show job_controller.channel_class:ims-ms*.master_command
role.job_controller.channel_class:ims-ms*.master_command = IMTA_BIN:ims_master
```

But note that the `ims_master` channel program has two optional switches.

- `-d` specifies that minimal debugging is enabled (corresponds to setting the [DEBUG ims-ms-channel-specific option](#) to a value of `1`; a higher `DEBUG` option value will override this command line effect).
- `-t num-threads` specifies that `num-threads` should be used; the [DELIVER_THREADS ims-ms-channel-specific option](#), if specified, will override this value.

51.4 ims-ms channel debugging

ims-ms channel debugging goes to two (or three) places, the `channelname_master.log-*` file, and the `imta` file, which is an NSLOG file, and as of JES MS 6.2, if message tracing is enabled (if the [message trace.activate](#) option in Unified Configuration or the `local.msgtrace.active` configutil parameter in legacy configuration is set to `yes`), then message tracing will also be written to the `msgtrace` file.

MTA processing debugging, enabled for instance via the `master_debug` channel keyword, goes to an `channel-name_master.log-*` file.

ims-ms channel level processing goes to the `imta` file. In general, the level of detail recorded in the `imta` file is controlled by the `mta.loglevel` option in Unified Configuration or the configutil parameter `logfile.imta.loglevel` in legacy configuration; it can be set to any of the values `critical`, `error`, `warning`, `notice`, `information`, or `debug`. In

order for the debug output generated due to a non-zero `DEBUG` `ims-ms-channel-specific` option setting (`ims-ms.options.DEBUG` in Unified Configuration, or `DEBUG` option in the `ims-ms` channel option file in legacy configuration) to in fact get included in the `imta` log file, `mta.loglevel` (Unified Configuration) or `logfile.imta.loglevel` (legacy configuration) must hence be set to `debug`.

Chapter 52 Other channels

52.1 Local channel	52-1
52.2 Bitbucket channel	52-2
52.2.1 Bitbucket channel configuration	52-2
52.3 Defragmentation channel	52-3
52.3.1 Defragmentation channel configuration	52-3
52.3.2 MAX_PARTS Defragmentation-channel-specific option	52-4
52.3.3 Defragmentation channel message retention time	52-4
52.3.4 Multi-host defragmentation channel operation	52-5
52.4 filter_discard channel	52-7
52.4.1 Retrieving messages from the filter_discard channel	52-8
52.5 Generic SMTP channels	52-9
52.6 Hold channel	52-10
52.6.1 Hold channel configuration	52-10
52.6.2 Releasing messages from the hold channel	52-11
52.6.3 Diagnosing .HELD files	52-11
52.7 Pipe channels	52-12
52.7.1 Setting up a pipe channel	52-13
52.7.2 Pipe entry match order	52-17
52.8 Process and Reprocess channels	52-18
52.8.1 Reprocess channel operation as prior channel	52-19

The MTA has a number of subsidiary channels, (that is, channels used for internal or special processing purposes, rather than for delivery to local mailboxes or relaying to remote hosts), and a sample "generic" SMTP channel provided as an example of channel code, including:

- the [bitbucket channel](#),
- the [conversion channel](#),
- the [defragment channel](#),
- the [filter_discard channel](#),
- the "generic" SMTP channels,
- the [hold channel](#),
- the [pipe channel](#),
- the [process and reprocess channels](#),

The [conversion channel](#), which has relatively complex configuration, is discussed separately under [Message conversions](#). The other channels listed are discussed in the chapter below.

52.1 Local channel

In modern usage, the local or "l" channel's main use is as a placeholder in identifying "local" addresses, in that (normally) only addresses that initially rewrite to the local channel will be checked for MTA [aliasing](#). (See, however, the [aliaslocal](#) channel option.) The presence of the [viaaliasrequired](#) channel option on the "l" channel is critical both operationally

and for security purposes, as it enforces that *only* those addresses that have a successful alias lookup (hence correspond to a provisioned address) are accepted as valid "local" addresses.

In principle, the local channel could also be used to deliver messages to UNIX mailboxes on the local host; however, that use is deprecated.

In principle, the local channel would also be considered to be the source of messages submitted on the iMS host itself via utilities such as `sendmail`, `mail`, `mailx`, `mailtools`, `imsimta send`, *etc.*; however, such submission is mostly moot nowadays.

Configuration settings on the "I" channel are somewhat privileged, since it is taken (in the absence of overriding configuration settings) as the default set of names for the MTA system itself. See the [notices](#) channel option. And compared to the "I" channel [official_host_name](#), see the [id_domain](#) and [received_domain](#) MTA options and the [BANNER_HOST](#) TCP/IP-channel-specific option for some examples of configuration options to override this "I" channel value.

52.2 Bitbucket channel

As the name itself suggests, the bitbucket channel simply deletes any message enqueued to it. Indeed, messages that match the bitbucket channel are instantly deleted, without even being written to a bitbucket disk area. (In particular, note that no `IMTA_QUEUE:bitbucket/*` (or equivalently `$SERVERROOT/data/queue/bitbucket/*`) message files are created---a message is simply discarded immediately once the MTA sees that it matches the bitbucket channel. However, if logging is enabled then the MTA does write [MTA transaction log entries](#) for the bitbucket channel as if it actually ran, for monitoring/statistics purposes. Note that the bitbucket channel "D" supposed "dequeue" record is in fact written by the enqueueing process at the same time as it writes its "E" supposed "enqueue" record: note the identical time stamps and, if the MTA option `log_process=1` is set, the identical process id corresponding to the enqueueing process.)

Note that [Sieve filter "discard" and "jettison" actions](#) are, depending upon the setting of the `filter_discard` and `filter_jettison` MTA options, potential sources of purported "enqueues" to the bitbucket channel, as are matches of an attempted poster's envelope From address to the value of an `mgrpJettisonDomain` or `mgrpJettisonBroadcasters` LDAP attribute (more precisely to the value of whatever LDAP attributes are named by the `ldap_jettison_domain` or `ldap_jettison_url` MTA options), as well as `*_ACCESS` [mapping table \\$V, \\$v, \\$Z, or \\$z](#) flag effects.

52.2.1 Bitbucket channel configuration

Bitbucket channel configuration is quite simple and minimal, consisting of a bare bones channel definition and a few rewrite rules to recognize pseudodomain name(s) which will be used when directing messages to the bitbucket channel. (Even if users never explicitly type in a bitbucket domain name, note that the MTA itself uses some bitbucket pseudodomain names internally, as in certain [delivery_options](#) values.)

For instance:

```
msconfig> show channel:bitbucket
role.channel:bitbucket.official_host_name = bitbucket-daemon
msconfig> show rewrite.rule * bitbucket*
role.rewrite.rule = bitbucket $U%bitbucket.domain.com@bitbucket-daemon
role.rewrite.rule = bitbucket.&/IMTA_HOST/ $U%bitbucket.domain.com@bitbucket-daemon
```

52.3 Defragmentation channel

The MIME standard ([RFC 2046](#)) 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. It can also be useful when sending over networks subject to connection drops, as a form of "check-pointing" of the sending of a message, since a connection drop during transmission of a fragment of a message will require resending only of that fragment, rather than of the entire message. Information is included in each part so that the message can be automatically reassembled once it arrives at its destination.

The `defragment` channel option (used on channels *other* than the defragmentation channel) and the defragmentation channel itself provide the means to reassemble messages in the MTA. When a channel is marked `defragment`, any message/partial messages queued to that channel will be placed in the defragmentation channel queue instead. The defragmentation channel maintains a database which is used to match the parts of each message up with each other. Once all the parts have arrived, the message is rebuilt and sent on its way.

The defragment database can optionally be stored on a filesystem accessible to multiple hosts (for instance, over NFS), and then shared by multiple hosts. Such sharing of the defragment database can be particularly useful for achieving appropriate and efficient message defragmentation in multi-tiered, multi-plexed deployments. See [Multi-host defragmentation channel operation](#) for further details.

All channels that perform local delivery or send messages on to hosts (*e.g.*, LMTP back end Message Store hosts) or networks that cannot deal with fragmented messages should be marked with the `defragment` channel option. In particular all `ims-ms` and `tcp_lmtpcs*` channels should be marked with the `defragment` channel option. The `defragment` channel option will have no effect unless a defragmentation channel is also defined.

A defragmentation channel is produced automatically by the MTA configuration generator.

52.3.1 Defragmentation channel configuration

A defragmentation channel is normally generated as part of an initial configuration.

In legacy configuration, the defragmentation channel definition consists of a channel entry, at its most minimal perhaps merely:

```
defragment
defragment-daemon
```

or as of 8.0:

```
defragment receivedstate "convert/defragment"
defragment-daemon
```

and rewrite rules of the form:

```
defragment                $U@defragment.localhostname@defragment-daemon
defragment.localhostname  $U@defragment.localhostname@defragment-daemon
```

where `localhostname` should be replaced by the name of the local host.

In Unified Configuration, the equivalent would be:

```
msconfig> show channel:defragment.*
role.channel:defragment.official_host_name = defragment-daemon
msconfig> show rewrite.rule * defragment*
role.rewrite.rule = defragment $U%defragment.domain.com@defragment-daemon
role.rewrite.rule = defragment.&/IMTA_HOST/ $U%defragment.domain.com@defragment-daemon
```

(with `domain.com` being replaced by a site's own domain name).

Once such a defragment channel and rewrite rules are in the configuration, then an address of the form

```
user%host@defragment.localhostname
```

will be routed through the defragmentation channel. (Sending anything other than a message/partial message to the defragmentation channel causes the channel to simply requeue the message for normal delivery.)

52.3.2 Defragmentation-channel-specific option: MAX_PARTS (10 <= integer <= 100,000)

New in Messaging Server 7.4-18.01, the defragmentation channel supports, (in addition to the usual [channel options](#)), one defragmentation-channel-specific option, `MAX_PARTS`. `MAX_PARTS` specifies the maximum number of fragments a message can be broken into and still be reassembled. The maximum is 100,000; the minimum is 10; the default is 1000.

This defragmentation-channel-specific option would be set in legacy configuration in a channel-specific option file, `IMTA_TABLE:defragment_option`, or in Unified Configuration is set under the channel's `options` option. For instance:

```
msconfig> set channel:defragment.options.MAX_PARTS 500
```

Note that in Unified Configuration such channel-specific options are not (currently) schema checked: be careful when setting them as `msconfig` will not warn of invalid values or syntax as it would for regular channel options! (Nor will `msconfig` show whether or not such channel-specific options have any default value.)

In legacy configuration, where an option file is used, such an option file must be named `defragment_option`, and stored in the MTA table directory.

52.3.3 Defragmentation channel message 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 elapsed¹ (and a [backoff](#) time has elapsed so that the channel is attempting to process relevant message fragments), the various parts of a message will be sent on without being reassembled. This choice of time value (normally) eliminates the possibility of a nondelivery notification being sent about a message in the defragment channel queue.

The [notices](#) channel option (and optionally its priority-sensitive variants, [*notices](#)) controls the amount of time that can elapse before nondelivery notifications are sent, while the

`backoff` channel option (and optionally its priority-sensitive variants, `*backoff`) controls when message delivery attempts are made. So between the two of them, they control the amount of time messages are retained before being sent on in pieces, with `notices` normally being the primary control (controlling how long before the channel "gives us" on attempting to reassemble message fragments), modulated by when the `backoff` values in fact cause the channel to run and attempt message delivery (hence deliver message fragments that have lingered past half the final `notices` value and still do not make up a complete message "as is", without reassembly). Set the `notices` option value to twice the amount of time you wish to retain messages for possible defragmentation (and ensure that the `backoff` values in effect for the defragment channel are such that a delivery attempt will be made shortly after half the final `notices` value). For example, normally a `notices` value of 4 would cause retention of message fragments for two days:

```
defragment notices 4
defragment-daemon
```

Or if the more general `backoff` values do not align well with the defragment channel's `notices` time scale, then also set some sensible `backoff` values explicitly on the defragment channel. For instance, setting the final `backoff` value to `p4h` means that message (fragments) will get a reassembly-and-possible-delivery attempt every four hours. Hence the following sort of definition:

```
defragment backoff "p1h" "p2h" "p2h" "p4h" notices 4
```

will mean that message fragments will get sent on as is (still as fragments, rather than reassembled) after approximately forty-nine hours. (There will be message delivery attempts immediately, then at one hour, three hours, five hours, nine hours, then every four hours thereafter---so the first time past the two day point, the expiration of half of the `notices` final value, will be at forty-nine hours.)

When the defragment channel cannot reassemble a message, and decides to send onwards a message fragment, it will add a header line

```
Defragment-failed: official-local-host-name
```

showing the official local host name from the L channel.

¹ Note that the "half" is calculated using integer integral division but not allowed to go below one, so for instance an initial `notices` value of 5 results in fragments being eligible for delivery after 2 days or hours depending upon `return_units`, but even an initial `notices` value of 1 still means fragments aren't eligible for delivery until after 1 day or hour.

52.3.4 Multi-host defragmentation channel operation

The defragment database can optionally be stored on a filesystem accessible to multiple hosts (for instance, over NFS¹), and then shared by multiple hosts. This can be particularly useful when multiple "front end" hosts can potentially deliver to the same "back end" message stores, particularly when the "back end" message store can not do message defragmentation itself (as for LMTP message stores).

To set up such sharing, make a link from the `msg-root/config/defragment_cache` on each individual system to whatever file you want to have be the shared defragment

database on the shared (NFS) disk. Note that the NFS mounted file system should be set for "soft mount" with a relatively short time out, rather than for (the default for NFS) "hard mount". Regarding the NFS time out, the NFS mount option (see the `mount_nfs(1M)` man page) `timeo` will need to be set on the `/etc/dfs/dfstab` entry (or `automount map`) that causes the file system to be mounted. (With a "hard mount", if NFS went down then the defragment channel would hang, waiting for the access to the defragment cache to succeed. But with a "soft mount", the defragment channel will time out its attempt to access the defragment cache. So the channel will not hang; instead, in the unlikely event that all message fragments happen to end up on one host, that host's defragment channel should be able to reassemble the fragments and send the message onwards, properly reassembled; but more likely, the fragments will be spread among different hosts, none of which can reassemble or properly route to another host's defragment channel in the absence of successful access to the defragment cache, so instead the various fragments will eventually get sent onwards still as separate fragments.)

When setting up a defragment database that will be shared over NFS by multiple systems, note that the [MTA user](#) (typically `mailsrv` -- see the `user` option in `restricted.cnf`) on each system must be defined to have the same `uid` number on each system. If systems define the MTA user with different `uid` numbers, permission problems can be expected.

The defragment database entries include a field specifying the host upon which a message fragment resides. Once an initial part 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 that "first" host that received the "first" part. (The defragment channel when it runs, first checks if any message fragment parts are already present, and if so on which host; then if a part or parts are already present on some other host, the defragment channel sends its just-received part onward to the other host, using explicit source routing to route to the other host, rather than retaining the part for reassembly attempts itself. See the [Multi-host defragmentation channel operation example](#) for an example.) Thus all remaining parts of a fragmented message end up getting redirected to the host whose defragment channel happened to attempt processing the very first (first to arrive, not necessarily `part=1`) part of the message; that host's defragment channel is then responsible for doing the message defragmentation (reassembly) once all fragments have been received. (One consequence is some load-balancing of the defragmentation of messages depending upon which host happens to receive the "first" part of each message.)

¹ Note that sharing the defragmentation database over NFS is an exception to the general rule that the MTA does not support sharing filesystems via NFS. The MTA's use of the defragment database has been specially designed with NFS' limited locking semantics in mind.

52.3.4.1 Multi-host defragmentation channel operation example

The [defragmentation channel](#) supports operating in a setup where [multiple hosts share the same defragmentation database](#). In such setups, the routing of message fragments to and between defragment channels on a multi-host setup (when multiple hosts are sharing the same defragment database) is as follows:

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. The defragment channel on host 1 runs, checks the defragment database for whether any other parts of this message have yet arrived. None have, so 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. The defragment channel on host 2 runs, checks the defragment database, sees that part x of this message is already present and stored on host 1. So the host 2 defragment channel redirects the message over 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 host 1 defragment channel runs and enters it into the database, *etc.*
6. Eventually, the "final" fragment of the message gets to host 1 (possibly having first gone through another host such as host 2, which then routed the message to host 1). The defragment channel runs and now sees that it has all the fragments of the message, reassembles them all into the original message, and sends the message onwards to the intended recipient.

52.4 filter_discard channel

By default, messages [discarded via a Sieve filter](#) are immediately discarded (deleted) from the system. However, when users are first setting up Sieve filters (and perhaps making mistakes), or for debugging purposes, it can be useful to have the deletion operation delayed for a period. (Certain flags in [*_ACCESS mapping tables](#) cause Sieve filter like discarding of messages; such messages also are eligible for delayed deletion.)

To have Sieve filter discarded messages temporarily retained on the MTA system for later deletion, first add a `filter_discard` channel in your MTA configuration, *e.g.*:

```
filter_discard notices 7
FILTER-DISCARD
```

or in Unified Configuration:

```
msconfig> set role.channel:filter_discard.official_host_name FILTER-DISCARD
msconfig# set role.channel:filter_discard.notices 7
```

As of 8.0, use of the new-in-8.0 [receivedstate](#) channel option is recommended, so:

```
msconfig> set role.channel:filter_discard.receivedstate "quarantine/sieve-discarded"
```

or in legacy configuration:

```
filter_discard receivedstate "quarantine/sieve-discarded" notices 7
FILTER-DISCARD
```

with the [notices](#) channel option specifying the length of time (normally number of days) to retain the messages before deleting them. Then set the MTA option [filter_discard=2](#):

```
msconfig# set role.filter_discard 2
```

By default, messages discarded due to a Sieve "jettison" action get the same handling as those discarded due to a "discard" action, as controlled by the `filter_discard` MTA option, either being deleted from disk immediately or retained in the `filter_discard` channel queue area. However, the `filter_jettison` MTA option may be used to differentiate the handling; for instance, if one wishes to retain messages discarded by a (presumably user level) "discard" action in the `filter_discard` channel, while immediately deleting messages discarded due to a (system level) "jettison" action, one could set `filter_discard=2` and `filter_jettison=1`:

```
msconfig> set role.filter_discard 2
msconfig# set role.filter_jettison 1
```

Setting the MTA option `filter_jettison=2` explicitly (or the implicit effect if the MTA option `filter_discard=2` is set) will cause messages discarded due to a Sieve "jettison" action to be retained in the `filter_discard` channel.

Messages in the `filter_discard` channel 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` channel 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` or the `imsimta qm` utility's `return` command.

52.4.1 Retrieving messages from the filter_discard channel

One reason for configuring use of the `filter_discard channel` (rather than simply having "discarded" messages immediately deleted from disk) is that it permits the potential for the system administrator to "retrieve" messages while they are still in the `filter_discard` channel queue on disk. The general process for such "retrieval" is as follows:

1. Move the message files in question manually to the `reprocess channel` queue area (typically `server-root/data/queue/reprocess`), changing the filename slightly (e.g., from `*.00` to `*.02`) when doing so, and making sure to preserve the proper ownership and protections on the message files.
2. Issue the command `"imsimta cache -synch"` so that the `Job Controller` will do an immediate scan to notice the moved message files. (If one is in no particular hurry for such messages to begin getting delivery attempts, this step can be omitted in favor of waiting for the `Job Controller` to get around to noticing such files on its own; the `Job Controller` normally scans automatically every four hours per its `synch_time` option.)
3. After the `imsimta cache -synch` has completed, one may issue the command `"imsimta run reprocess "` to `run` an extra `reprocess` channel job immediately, rather than waiting for the `Job Controller` to get around to scheduling a `reprocess` job to process any messages (such as the recently moved messages) that it might have due for processing.

When the MTA applies a "discard" or "jettison" action, a bit in the message envelope gets set; if the MTA subsequently sees a message file that has that bit set, it will not again apply a "discard" or "jettison" action. The purpose of this is precisely so that the above sort of

operation (moving a previously discarded message from the `filter_discard` channel to the [reprocessing channel](#)) can be used to get messages delivered, without the "discard" getting re-applied.

Note that the discarding of a message can be caused by a "discard" or "jettison" actions arising from any of a number of locations, including a number of potentially applicable Sieve filters (system, channel, user, group, domain, *etc.*), or by address-based [*_ACCESS mapping table](#) `$v`, `$V`, `$z`, or `$Z` flags. Wherever/whatever the source of the discarding action, the setting of the same ignore-future-discard-actions bit occurs, and if such a message is subsequently "retrieved" (reinjecting into the MTA's regular channel queues typically being moved manually to the reprocess channel queue area), then *all* such discarding actions that might otherwise be applicable on this MTA will be henceforth ignored for the message in question (on this MTA -- such delivery flags are not normally transferred to other MTAs, though see the [flagtransfer](#) channel option).

One more note: messages enqueued to the `filter_discard` channel always get the username field for the message (the "owner" of the message for access purposes) set to the string "FILTER_DISCARD"; this value will, for instance, show up in the `log_username` field of the [MTA message transaction log file](#).

52.5 Generic SMTP channels

The channel programs `test_smtp_master` and `test_smtp_slave` are provided as models upon which additional channels using the SMTP protocol can be built. They are intended as examples only and not as production channel programs.

Both programs require that the environment variable `PMDF_CHANNEL` (on UNIX) translate to the name of the channel they are servicing -- and expect that channel to be defined in the MTA configuration.

When `test_smtp_master` is executed, it asks the [Job Controller](#) for messages waiting to be processed by the channel `PMDF_CHANNEL`. On UNIX, SMTP commands are written to `stdout` and responses are expected on `stdin`.

Similarly, on UNIX, `test_smtp_slave` accepts SMTP commands on `stdin` and writes responses to `stdout`.

The `imsimta run` utility and the regular configuration of the [Job Controller](#) never invoke `test_smtp_master` and will have to be modified in order to use `test_smtp_master`. The configuration to execute `tcp_master` can be used as a model to drive `test_smtp_master`.

`test_smtp_master` includes code to distinguish between use as a direct connection to the target system and use for routing through a gateway. This facility parallels the gateway support found in [TCP/IP channels](#), namely support for the `daemon` option.

Though `test_smtp_master` and `test_smtp_slave` never open or receive, respectively, an actual TCP/IP connection, if the environment variables `TRANSPORTINFO` and `APPLICATIONINFO` are set, then these programs will use that information to initialize the relevant fields that would be present in a real SMTP-over-TCP/IP message transport.

`test_smtp_master` and `test_smtp_slave` perform normal MTA channel initialization steps, including consulting the MTA configuration to determine if the named channel (the channel `PMDF_CHANNEL` translates to) has any [local_host_alias](#) set; they check for any [TCP/IP-channel-specific options](#); and they "support" typical channel options relevant to SMTP-

over-TCP/IP channels, such as `master_debug`, `slave_debug`, `smtp*`, and (as previously mentioned) the `daemon` channel option.

52.6 Hold channel

When messages are incoming to a user whose `mailUserStatus` attribute (more precisely, the attribute named by the `ldap_user_mail_status` MTA option) is set to a value of `hold`, or who is in a domain whose `mailDomainStatus` attribute (more precisely, the attribute named by the `ldap_domain_attr_mail_status` MTA option) is set to a value of `hold`, then the messages will be routed to the hold channel and will be marked as `.HELD`. (Both of these things---the routing to the hold channel, and the marking of the message as `.HELD`---are configured via the `delivery_options` MTA option's value for the `hold` clause.)

The hold channel is intended for temporarily detaining messages addressed to users (or to users in domains) for purposes such as migration of the users' (or an entire domain of users') mailboxes. The incoming messages can be sidelined in the hold channel queue area on disk, and then once the users' mailboxes are once again ready to receive e-mail, and the user and/or domain status has been set back to `active`, the messages can be delivered onwards.

[Hold channel configuration](#) discusses the basic configuration of the hold channel. [Releasing messages from the hold channel](#) discusses how to release messages from the hold channel.

52.6.1 Hold channel configuration

The initial configuration generated during an install generates a hold channel. The channel definition and basic rewrite rules would normally appear as

```
msconfig> show channel:hold
role.channel:hold.official_host_name
msconfig> show rewrite * *hold*
role.rewrite.rule = hold-daemon $U%H@hold-daemon
role.rewrite.rule = .hold-daemon $U%H@hold-daemon
```

Or in legacy configuration:

```
hold
hold-daemon
```

and rewrite rules appearing as

```
hold-daemon $U%H@hold-daemon
.hold-daemon $U%H@hold-daemon
```

Note that as of JES MS 6.0, the hold "channel" is in fact merely a variant (same image, but run as a different channel) of the [reprocess channel](#). In particular, the [Job Controller](#) configuration should include:

```
msconfig> show job_controller.channel_class:hold
role.job_controller.channel_class:hold.master_command = IMTA_BIN:reprocess
```

Or in legacy configuration, the Job Controller configuration file, normally `job_controller.cnf`, should contain a definition:

```
[CHANNEL=hold]
master_command=IMTA_BIN:reprocess
```

52.6.2 Releasing messages from the hold channel

To cause messages sidelined as `.HELD` in the hold channel queue area to become eligible for delivery (to be delivered onwards) once again, set the [user's status](#) (`mailUserStatus`) and [domain status](#) (`mailDomainStatus`) once again to `active`, and then use the `imsimta qm` utility's `release` command to release the held messages for processing. As of Messaging Server 7.0u4, the `imsimta qm` utility informs the [Job Controller](#) of such message release, which for normal messages usually results in an "immediate" (more-or-less, depending on current load on the MTA) delivery attempt. (But in some earlier versions, the released message would not get "seen" by the Job Controller until the next `cache -sync` occurred, whether via an explicit `imsimta cache -sync` command, or via the Job Controller's [own automatic, periodic cache -sync](#).)

Note that attempting to release a message for a user who is still marked as "hold" will merely result in the message again getting sidelined as `.HELD` with a temporary error (recorded in the [MTA message transaction log](#), and the message's own delivery history) using the [error_text_still_held](#) MTA option's text, so by default

```
452 4.2.1 cannot reenqueue while still held
```

52.6.3 Diagnosing .HELD files

The causes of `.HELD` files can be considered to fall into three major categories:

1. Messages `.HELD` due to a [user status](#) or [domain status](#) of "hold": These are messages that are, by intent of the MTA administrator, intentionally being side-lined, typically while some maintenance procedure is being performed, (*e.g.*, while moving user mailboxes).
2. Looping messages: messages that the MTA side-lined as `.HELD` because the MTA detected (via [build-up of one or another sort of *Received: header lines](#)) that the messages were looping.
3. Suspicious messages: messages that met some suspicion threshold, and were therefore side-lined as `.HELD` for later, manual inspection and action by the MTA administrator. Messages can be side-lined as `.HELD` due to exceeding a configured maximum number of envelope recipients (see the [holdlimit](#) channel option), due to exceeding [max_mime_levels](#) or [max_mime_parts](#) if such an MTA option has been set to a negative integer, due to an MTA administrator `imsimta qclean`, `imsimta qm clean` or `imsimta qm hold` command executed by the MTA administrator based on some suspicion of the message(s) in question, or due to use of a ["hold" action in a Sieve script](#).

When diagnosing the reason why a particular message or messages are `.HELD`, consider the following:

1. All messages `.HELD` due to a [user status](#) 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.

2. Messages .HELD due to the MTA having detected that they were looping can be expected to have a great many of one or another sort of *Received: header lines, when inspected. Furthermore, those Received: header lines typically illustrate the exact path of the message loop: look especially carefully at the hostnames and any recipient address information (*e.g.*, "for recipient" clauses or "(ORCPT recipient)" comments) appearing in such header lines. One cause of such message loops is user error: a user forwards their messages on system A to system B, and has system B set up to forward back to system A. (The solution is for the user to fix their forwarding definitions.) 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"; more on this below). Or another possible cause is missing or erroneous host table entries or DNS records that cause routing back to the MTA of addresses in domains not intended to be handled by this MTA. (The solution in such a case should be to correct the name records at the TCP/IP level; when that can not be achieved, *some* host table or nameserver problems can be worked around by special routing configuration on the MTA, though solving any underlying name problems is preferable.) Note that the MTA's thresholds for determining that a message is likely "looping" are configurable; see [Received header line MTA options](#). Also note that the MTA may optionally be configured---see the [held_sndopr](#) MTA option---to generate a syslog notice whenever a message is forced into .HELD state due to exceeding such a threshold. If syslog messages of

HELDMSG, Header count exceeded; message has been marked .HELD automatically
are present, then you know that this is occurring.

3. Messages .HELD due to some suspicious characteristic will of course exhibit that characteristic---through *a priori* that characteristic could be anything which the site has chosen to characterize as "suspicious". If you are an administrator of the MTA, you should attempt to stay reasonably aware of whatever such configuration choices and actions have been taken on your MTA. 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 option, any use of the \$H flag in the [FROM_ACCESS mapping table](#) or a [recipient-address-based *_ACCESS mapping table](#), any negative value for the [max_mime_levels](#) or [max_mime_parts](#) MTA options, or any use of the "hold" action in any system Sieve file (the [systemfilter](#) MTA option/system level `imta.filter` file, or any channel level Sieve filters configured and named via use of [sourcefilter or destinationfilter channel options](#)); and ask any fellow MTA administrators about any manual command line side-lining (via, for instance, a `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 in [MTA message transaction logging](#); see the [log_filter](#) MTA option. (As of MS 8.0 with its new, private Sieve "transactionlog" extension, MTA administrators may wish to make use of the "transactionlog" action in any Sieve filter that performs a "hold" action, to record details regarding the reason(s) that the "hold" was performed.)

52.7 Pipe channels

Pipe channels are used to effect delivery for specific addresses via a site-supplied program or script. While pipe channels are loosely based upon the | (pipe) functionality of sendmail,

they have been carefully designed to not pose a security threat. First, the MTA administrator must manually add a pipe channel to the configuration before it can be used. The MTA administrator may refrain from adding a pipe channel to the configuration, if he or she prefers not to trust the technology. Second, the commands executed by the pipe channel are controlled by the MTA administrator, and no user supplied input can find its way into those commands: each command the administrator supplies is used verbatim with the exception of the optional substitution of a filename generated by the channel itself without reference to user supplied input. Finally, the decision to run a command is not based upon the presence of special characters appearing in a recipient address, but rather upon a recipient address exactly matching a specific address or host name in a table or database which the MTA administrator must provide. If an exact match between an incoming address and the site's table or database exists, then the command listed in the table for that match is executed.

Unlike the sendmail pipe functionality, the MTA's pipe channel does not pipe the message to be processed to the program or script. Instead, it writes the message to be processed to a temporary file and then forks a subprocess to run the site-supplied command for that message. That command should make use of the name of the temporary file which can be substituted into the command by the channel. The temporary file should not be deleted or altered by the subprocess; the channel will delete it itself. If it is not possible to prevent the subprocess from disrupting the file, then the pipe channel should be marked with the `single` channel option.

On UNIX systems, if the subprocess exits with exit code of 0 (`EX_OK`) then the message is presumed to have been delivered successfully and is removed from the MTA's queues. If it exits with an exit code of 71, 74, 75, or 79 (`EX_OSERR`, `EX_IOERR`, `EX_TEMPFAIL`, or `EX_DB`) then 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 `sysexit.h`.

52.7.1 Setting up a pipe channel

There are two steps in setting up a pipe channel:

1. [adding the channel definition to the configuration](#), and
2. [specifying the addresses to receive pipe channel processing and their corresponding processing](#), via LDAP attributes on users, or via the channel option file, pipe database, or profile database entries

52.7.1.1 Adding a pipe channel to the configuration

Note: Usually a single pipe channel can suffice; but it is possible to configure multiple pipe channels, if desired.

A message to be processed by a pipe channel is usually routed to the channel via a combination of an alias and rewrite rules. For instance, the system domain.com might want all mail for the addresses `info-list@domain.com` and `gripes@domain.com` to be routed to a pipe channel. This could be accomplished with the alias file entries

```
info-list: info-list@pipe.domain.com
gripes: gripes@pipe.domain.com
```

where `pipe.domain.com` is in turn a host name associated with a pipe channel via rewrite rules. For instance,

```
pipe.domain.com          $u%pipe.domain.com@PIPE-DAEMON
```

So, to configure a pipe you need to determine the host names, `pipe1.domain`, `pipe2.domain`, ... which you wish to use. Once you have determined these, add them to the rewrite rules section of your MTA configuration file:

```
pipe1.domain            $u%pipe1.domain@PIPE-DAEMON
pipe2.domain            $u%pipe2.domain@PIPE-DAEMON
...                     ...
```

Then, to the end of your MTA configuration, add the definition of the pipe channel itself:

```
pipe
PIPE-DAEMON
```

Be sure to include a blank line *before and after* this channel definition.

On UNIX, the pipe channel runs by default as the MTA user, as specified by the `user` option in `restricted.cnf` (or the `imta_user` MTA Tailor option in versions prior to MS 7.0.5). So on UNIX, if you wish the pipe channel to run as some other user, you may use the `user` channel option to specify the desired username. Note that the argument to `user` is normally forced to lowercase, but original case will be preserved if the argument is quoted.

At this point, the pipe channel has been added to the configuration. However, it cannot be used until you create a channel option file, or a pipe database, or (UNIX only) define and set delivery methods for pipe channel addressees, as described next.

52.7.1.2 Pipe channel addressees and their handling

In order for a pipe channel to handle an address, each specific recipient address (or at least each domain) to be handled must be explicitly configured with the desired pipe channel handling. There are several ways to do this configuration.

Nowadays, most typically users to receive pipe channel handling would get [markings in their LDAP entries](#), and [Pipe options](#) would define the program(s) available to run to process these users.

Older approaches using the [MTA profile database](#) or [Pipe channel option file](#) also exist.

52.7.1.2.1 User LDAP attributes for pipe channel processing

When the MTA finds a user entry in LDAP that has a `mailDeliveryOption` attribute (or whatever LDAP attribute is named by the `ldap_delivery_option` MTA option) value of program, the `delivery_option` MTA option's `program` clause defines this to mean (by default) to route the message to the pipe channel with the address composed of the user's `uid` value and the value of the user's `mailProgramDeliveryInfo` LDAP attribute (or whatever attribute is named by the `ldap_program_info` MTA option), *i.e.*, `uid@mailProgramDeliveryInfo`.

The definition of what that `mailProgramDeliveryInfo` value means is controlled by [Pipe options](#).

52.7.1.2.1.1 Pipe options

A few options control operation of the Pipe channel.

Note that the Pipe channel is used when a user's `mailDeliveryOption` LDAP attribute (or more precisely, the attribute named by the `ldap_delivery_option` MTA option) includes a value of `program`, with execution as defined via the MTA `delivery_options` option's `program` clause. The `delivery_options` option's `program` clause (normally) defines such program delivery to mean routing to the pipe channel with an address that includes the value of the user's `mailProgramDeliveryInfo` attribute (or whatever attribute is named by the `ldap_program_info` MTA option).

52.7.1.2.1.1.1 `params` Option

The `params` `Pipe` option specifies program delivery arguments. The substitution sequence `%s` may be used in the value to cause substitution of the (current) username.

52.7.1.2.1.1.2 `perms` Option

The `perms` `Pipe` option specifies permissions for the pipe delivery program.

52.7.1.2.1.1.3 `command` Option

The `command` `Pipe` option specifies the location of the program to execute to perform delivery.

52.7.1.2.2 Profile database

The Profile database is located at `IMTA_ROOT:data/db/profiledb`.

Entries in the Profile database are created and managed using the `imsimta profile` utility.

52.7.1.2.3 Profile database entries for pipe channel addressees

On UNIX, before looking in the `pipe database` or `pipe channel option file`, a pipe channel first queries the `MTA profile database` checking whether there is a delivery method set for the addressee. Only if there is no such entry is the pipe database or pipe option file consulted.

An MTA profile database delivery method entry for a pipe channel addressee is similar to that for a local (L) channel addressee, except that the pipe channel domain is used. For instance,

```
# imsimta profile
profile> set delivery MIME -user=jane.doe@pipe.domain.com
```

52.7.1.2.4 Pipe database

The pipe database was an alternate approach for storing larger numbers of address definitions than was convenient for the `Pipe channel option file` -- but this approach is now more-or-less obsolete.

52.7.1.2.5 Pipe channel option file

Unless you have a `pipe database`, or have established delivery methods for pipe channel addressees, each pipe channel must have an option file. If there are no delivery methods set in the MTA profile database, nor a pipe channel option file, nor a pipe database, then the channel will not operate.

The commands to execute for each envelope recipient address presented to the channel are specified in the MTA profile database, in the `pipe database` or in the pipe channel's option file.

If an address does not appear in one of these locations, then an error notification is sent back to the message originator.

In legacy configuration, pipe channel option files are stored in the MTA configuration directory `SERVERROOT/config/advanced` and have names of the form `x_option`, where `x` is the name of the pipe channel to which the option file applies. (In most instances, the file name will be `pipe_option`; *i.e.*, `SERVERROOT/config/advanced/pipe_option` on UNIX.)

To process the address `user@host`, the pipe channel first probes the option file for an entry of the form

```
user@host=command
```

If no matching entry is found, the channel next probes the option file for an entry of the form

```
host=command
```

If still no matching entry is found, then the recipient address is deemed bad and an error notification is sent back to the message originator. See [Pipe entry match order](#) below for an additional discussion of the order in which probes of various forms are made to the various possible entry sources.

If, however, a probe does find a matching entry, then the specified command, `command`, is executed. Prior to being executed, any occurrences of the phrase `%s` appearing in `command` are replaced with the name of the temporary file containing the message to be processed. It is important that the command to be executed neither delete nor otherwise alter the temporary message file as it may be needed for further pipe channel recipients of the same message. If disruption of the message file cannot be prevented, then mark the channel with the [single](#) channel option.

In Unified Configuration, the pipe channel option file is replaced by settings under `channel:pipe.options`, *e.g.*,

```
msconfig> set channel:pipe.options.user@host command
msconfig# set channel:pipe.options.host command
```

where note that any periods in the `host` will need to be backslash quoted. So for instance, if `host` is `mailhost.domain.com`, then the commands would appear as:

```
msconfig> set channel:pipe.options.user@mailhost\.domain\.com#command
msconfig# set channel:pipe.options.mailhost\.domain\.com command
```

The command to be executed will be run by a subprocess of the process running the pipe channel. As such, it will be running with the privileges of the user named as the pipe channel processing account via the `user` channel option on the pipe channel, or if that channel option was not used, then it will be running with the privileges of the MTA user account (see the `user` option in `restricted.cnf`). See [Pipe channels](#) for a description of the exit or completion codes with which the command should exit the subprocess.

Note: As with any MTA option file, it is important that the option file not be world writable. This is especially true of pipe channel option files.

In addition to the command entries in the pipe channel option file, there are additional general options available:¹

¹ General UNIX L channel/native channel options may also be specified in a pipe option file, though their relevance for the pipe channel tends to be limited.

52.7.1.2.5.1 SHELL_TIMEOUT (integer; UNIX only)

The SHELL_TIMEOUT option may be used to control how long in seconds the channel will wait for a shell command to complete. Upon such time outs, the message will be returned back to the original sender with an error message along the lines of "Timeout waiting for ...'s shell command ... to complete". The default value is 600 (corresponding to 10 minutes).

52.7.1.2.5.2 SHELL_TMPDIR (directory-specification)

The SHELL_TMPDIR option may be used to control where the pipe channel creates its temporary files when delivering to a shell command. By default, such temporary files are created in the home directory of the MTA user (or the user specified by the `user` channel option). Via this option the MTA administrator may instead choose to have the temporary files created in some (single) other directory; *e.g.* (legacy configuration):

```
SHELL_TMPDIR=/tmp
```

or (Unified Configuration):

```
msconfig> set channel:pipe.options.SHELL_TMPDIR /tmp
```

52.7.1.2.5.3 UNIX_STYLE (0 or 1)

Setting UNIX_STYLE=1 causes the pipe channel to write out a UNIX style mbox file, including a colonless From line, if a pipe command such as

```
cat %s >> filename.txt
```

is used. UNIX_STYLE=1 is the default on UNIX platforms. Setting UNIX_STYLE=0 tells the pipe channel not to write the colonless From line. *E.g.*, in Unified Configuration:

```
msconfig> set channel:pipe.options.UNIX_STYLE 0
```

52.7.2 Pipe entry match order

The logic for checking entries in the [profile database](#) (UNIX only), [pipe database](#) and [pipe channel option file](#) is as follows:

1. (UNIX only) Check the [profile database](#) for a `user@host` entry.
2. Check the [pipe database](#) for a `user@host` entry.
3. Check the [pipe option file](#) for a `user@host` entry.
4. Check the [pipe database](#) for a host entry.
5. Check the [pipe option file](#) for a host entry.

If no [profile database entry](#) exists and if the [pipe database](#) does not exist, then only the [pipe option file](#) is checked, *i.e.*, steps (3) and (5) only. If no profile database entry exists and if the pipe database exists but cannot be opened, then the message is passed over --- the condition is treated as a temporary error. And when checks (1)--(5) turn up no results, the message is bounced.

52.8 Process and Reprocess channels

The processing and reprocessing channels are essentially the intersection of all other channel programs --- they perform only those operations that are shared among all other channels. In other words, such a channel is simply a channel queue whose contents are processed and queued to other channels. Messages receive no special processing whatsoever.

The difference between a reprocessing channel and a processing channel is that a reprocessing channel is normally "invisible" as a source or destination channel, as for instance in a [CONVERSIONS](#), in a [CHARSET-CONVERSION](#), or in a [Recipient access mapping table](#) such as [SEND_ACCESS](#), or in a source channel or destination channel specific rewrite rule. A processing channel, on the other hand, is visible like other MTA channels.

It may appear that such a channel is effectively useless, but this turns out to be untrue. For example, the act of processing message bodies or generating message files for a message with a large number of recipients may be very time-consuming. Timeouts may occur if this is done during the operation of a channel slave program with an open network connection. So the MTA provides the [expandlimit](#) channel option, which forces queuing of the message to the reprocessing channel. Address expansion is then done as the reprocessing channel runs, free of any network timing constraints. The reprocess channel is also normally used to achieve deferred ("off-line") processing of expansion of mailing lists (see the [defer_group_processing](#) MTA option), and implementation of Sieve "redirect" actions; and in cases of LDAP directory unresponsiveness, submissions from "local" users are normally deferred to the reprocess channel (see the [domain_failure](#) MTA option). As of JES 5, a spam/virus filter package becoming unresponsive also optionally may cause such deferral; see the [spamfilterN_optional](#) MTA options.

If a message destined to an address of the form `user@domain` is routed to the reprocessing channel, (*e.g.*, due to rewrite rules or the [expandlimit](#) keyword) then the reprocessing channel will simply re-queue the message to the channel associated with the domain `domain`; if a message destined to an address of the form `user@reprocessing-domain` is routed to the reprocessing channel, (*e.g.*, as may be the case for mailing lists using deferred expansion), then the reprocessing channel will re-queue the message to the local channel. In either case, the reprocessing channel performs any necessary expansion of the `user` part of the address.

When an MTA channel has to generate a notification (bounce) message, such a notification message is initially enqueued to the processing channel.

A processing channel and a reprocessing channel are produced automatically by the MTA configuration generator. Note: Furthermore, for its own uses, the MTA will act as if process and reprocess channels are defined even if they are not explicitly present in your configuration. But if you wish to make any site specific uses of such channels, explicitly addressing or rewriting to such channels, then you will need to have the channels explicitly present in your configuration.

Prior to the 8.0 release, the `reprocess` channel handled [Sieve "redirect" messages](#); as of 8.0, Sieve "redirect" messages instead go through the [process channel](#).

52.8.1 Reprocess channel operation as prior channel

The `reprocess` channel performs various checks and operations as if it "were" the prior channel (the channel that enqueued to the `reprocess` channel); this includes some of its logging, as discussed in [Reprocess channel message transaction log entries](#). In particular, this operation-as-prior channel includes determination of whether [channel debugging](#) is enabled when the `reprocess` channel runs: the prior channel must have debugging enabled to obtain debugging for the `reprocess` channel. This operation as prior channel also includes the [address-based *_ACCESS mapping table](#) checks, where the probes are done "as if" the prior channel were probing; though note that as of 8.0, the [\\$:R input flag check](#) may be used to detect cases of the `reprocess` channel probing an address-based *_ACCESS mapping table.

Note that as of 7.0.5, [transactionlimit](#) settings from the prior channel are *not* applied when the `reprocess` channel is running.

52.8.1.1 Reprocess channel transaction log entries

Note that [MTA message transaction](#) (`mail.log*`) records involving the `reprocess` channel require a bit of special reading, as one of the whole goals of use of `reprocess` is to preserve the "original" source channel name (for purposes of access checks, *etc.*), and this then affects the message transaction log entries. For instance, in the case of a message enqueued (received) by the `tcp_intranet` channel and automatically deferred to the `reprocess` channel (perhaps because, say, the LDAP server is not currently responding, or because a Sieve "redirect" must be performed), relevant MTA message transaction log entries, with the MTA option [log_process=1](#) set, might have the rough form:

```
date-time-1 SMTP-process-id tcp_intranet reprocess E ...
date-time-2 reprocess-id    tcp_intranet tcp_local E ...
date-time-3 reprocess-id    reprocess      D ...
```

That is, the `reprocess` channel's enqueue onwards is recorded (and access checked, if appropriate) as if it were an enqueue from the original enqueueing channel---but if you have [log_process=1](#) set, then you can see by the process-id that it's actually the `reprocess` channel that did that second enqueue.

Prior to 8.0, the `reprocess` channel handled Sieve "redirect" messages; as of 8.0, Sieve "redirect" messages instead go through the [process channel](#).

Chapter 53 SMS options

53.1 SMS gateway options	53-2
53.1.1 enable Option Under sms_gateway	53-2
53.1.2 debug Option Under sms_gateway	53-2
53.1.3 foreground Option	53-3
53.1.4 history_file_directory Option	53-3
53.1.5 history_file_mode Option	53-3
53.1.6 history_file_flush_period Option	53-3
53.1.7 history_file_flush_threshold Option	53-3
53.1.8 history_file_rollover_period Option	53-3
53.1.9 max_conns Option Under sms_gateway	53-4
53.1.10 record_lifetime Option	53-4
53.1.11 thread_count_initial Option	53-4
53.1.12 thread_count_maximum Option	53-4
53.1.13 thread_stack_size Option	53-4
53.2 SMS gateway_profile options	53-4
53.2.1 text_to_subject Option	53-4
53.2.2 mta_channel Option	53-5
53.2.3 email_body_charset Option	53-5
53.2.4 email_header_charset Option	53-5
53.2.5 from_domain Option	53-5
53.2.6 in_re Option	53-5
53.2.7 parse_re_N SMS options	53-6
53.2.8 profile Option	53-7
53.2.9 route_to Option	53-8
53.2.10 select_re Option	53-8
53.2.11 smsc_default_charset Option	53-8
53.2.12 use_sms_priority Option	53-8
53.2.13 use_sms_privacy Option	53-9
53.3 SMS smpp_relay options	53-9
53.3.1 backlog Option Under smpp_relay	53-9
53.3.2 listen_addresses Option Under smpp_relay	53-9
53.3.3 listen_receive_timeout Option	53-9
53.3.4 listen_transmit_timeout Option	53-9
53.3.5 make_source_addresses_unique Option	53-9
53.3.6 max_conns Option Under smpp_relay	53-10
53.3.7 server_host Option Under smpp_relay	53-10
53.3.8 server_port Option Under smpp_relay	53-10
53.3.9 server_receive_timeout Option	53-10
53.3.10 server_transmit_timeout Option	53-10
53.3.11 tcp_ports Option Under smpp_relay	53-10
53.4 SMS smpp_server options	53-10
53.4.1 backlog Option Under smpp_server	53-10
53.4.2 esme_address_npi Option	53-11
53.4.3 esme_address_range Option	53-11
53.4.4 esme_address_ton Option	53-11
53.4.5 esme_password Option	53-12
53.4.6 esme_system_id Option	53-12
53.4.7 esme_system_type Option	53-12
53.4.8 listen_addresses Option Under smpp_server	53-12
53.4.9 listen_receive_timeout Option	53-13

53.4.10 `listen_transmit_timeout` Option 53–13
 53.4.11 `max_conns` Option Under `smpp_server` 53–13
 53.4.12 `server_host` Option Under `smpp_server` 53–13
 53.4.13 `server_port` Option Under `smpp_server` 53–13
 53.4.14 `system_id` Option 53–13
 53.4.15 `tcp_ports` Option Under `smpp_server` 53–13

SMS (Short Message Service) configuration allows some [global settings under `sms_gateway`](#), and below that is controlled via three named groups of options:

- General/global [SMS gateway options](#),
- named groups of [SMS gateway profile options](#),
`sms_gateway.gateway_profile:profile-name.option-name`
- named groups of [SMS `smpp_relay` options](#), `sms_gateway.smpp_relay:profile-name.option-name`, configuring aspects of the SMPP (Short Message Peer-to-Peer) relay operation, and
- named groups of [SMS `smpp_server` options](#), `sms_gateway.smpp_server:profile-name.option-name`, configuring aspects of the SMPP (Short Message Peer-to-Peer) server operation.

53.1 SMS gateway options

The SMS options set directly under `sms_gateway` establish general values, and global defaults. In particular, the [enable](#) option enables the SMS gateway on `start-msg` startup.

53.1.1 enable Option Under `sms_gateway`

The `enable` [SMS gateway option](#), `sms_gateway.enable` (Unified Configuration) or `local.msggateway.enable` (legacy configuration), enables the SMS service on `start-msg` startup.

53.1.2 debug Option Under `sms_gateway`

The `sms_gateway.debug` option enables debug output for the [SMS gateway](#). The default value is 6, which selects warning and error messages. The actual value of this option is a bit mask with the values shown below.

Table 53.1 SMS Gateway Debug Bit Mask Values

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.

53.1.3 foreground Option

The foreground [SMS gateway option](#), if set, means to run the SMS Gateway Server in the foreground with debugging enabled. See also the [sms_gateway.debug](#) option.

53.1.4 history_file_directory Option

The `history_file_directory` SMS gateway option specifies the 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 `IMTA_ROOT: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 SMS Gateway Server Storage Requirements to change this option to a more appropriate value.

53.1.5 history_file_mode Option

The `history_file_mode` [SMS gateway option](#) specifies permissions for files of historical data. The specified value is interpreted as an octal value.

53.1.6 history_file_flush_period Option

The `history_file_flush_period` [SMS gateway option](#) specifies how frequently cached history data is periodically flushed to disk; *i.e.*, the data is held solely in memory for no longer than `history_file_flush_period` milliseconds. By default, a value of 100 milliseconds (0.1 second) is used. Note that data written to disk is typically kept cached in memory as well. The data is written to disk so as to not lose it across server restarts.

53.1.7 history_file_flush_threshold Option

When the amount of cached history data exceeds the value of the `history_file_flush_threshold` [SMS gateway option](#), the cached data is written to the history file. By default, a value of 16M is used.

53.1.8 history_file_rollover_period Option

The current history file is closed and a new one created every `history_file_rollover_period` seconds. By default, a value of 3600 seconds (60 minutes) is used.

53.1.9 max_conns Option Under sms_gateway

The `max_conns` [SMS gateway option](#), `sms_gateway.max_conns`, specifies 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; see the `smpp_server.max_conns` and `smpp_relay.max_conns` values.

53.1.10 record_lifetime Option

The `record_lifetime` [SMS gateway option](#) specifies the 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 `history_file_rollover_period` seconds. Files on disk are purged when it becomes necessary to open a new history file.

53.1.11 thread_count_initial Option

The `thread_count_initial` [SMS gateway option](#) specifies the number of worker threads to initially create upon startup. By default, ten worker threads are initially created. The number of threads dynamically adjusts as the server runs.

53.1.12 thread_count_maximum Option

The `thread_count_maximum` [SMS gateway option](#) specifies the maximum number of concurrent worker threads to allow. By default, a maximum of fifty worker threads are allowed.

53.1.13 thread_stack_size Option

Each worker thread is outfitted with a stack whose maximum size is specified with the `thread_stack_size` [SMS gateway option](#). By default, a stack size of 64K is used. Specify a value of zero to use the platform's default thread stack size which is usually documented under or within the platform's `pthread_create` documentaton. Any non-zero value specified with this option will be rounded up to the nearest multiple of 64K which is not less than the operating system's minimum thread stack size.

53.2 SMS gateway_profile options

53.2.1 text_to_subject Option

By default, the content of an SMS message, when gatewayed to e-mail, is split between the Subject: header line and body of the resultant e-mail message, as per the setting of the

`parse_re_N` options. When the value of the `text_to_subject` [SMS gateway profile option](#), `gateway_profile.text_to_subject`, is set to 1, then the entire SMS text message is placed in the Subject: header line of the resulting e-mail message. No portion of the SMS text message is placed in the message's body.

53.2.2 mta_channel Option

The `mta_channel` [SMS gateway profile option](#), `gateway_profile.mta_channel`, specifies the name of the MTA channel used to enqueue e-mail messages. If not specified, then `sms` is assumed. The specified channel must be defined in the MTA's configuration.

For discussion of MTA channel definitions, (in particular, for discussion of Unified Configuration options set under a channel group), see instead [Channel configuration](#).

53.2.3 email_body_charset Option

The `email_body_charset` [SMS gateway profile option](#), `gateway_profile.email_body_charset`, specifies the character set to which to translate the SMS text prior to insertion into an e-mail 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 file `charsets.txt`.

This option is ignored when placement of the entire SMS message into the Subject: header line is enabled with the `text_to_subject` [SMS gateway profile option](#).

53.2.4 email_header_charset Option

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.

53.2.5 from_domain Option

The `from_domain` [SMS gateway profile option](#), `gateway_profile.from_domain`, specifies the domain name to append to SMS source addresses when constructing envelope From addresses for e-mail messages. The name specified should be the correct name for routing e-mail 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 `mta_channel` [SMS gateway profile option](#) (`CHANNEL` in legacy configuration) will be used.

53.2.6 in_re Option

The `in_re` [SMS gateway profile option](#) specifies the prefix text to use in the Subject: line of a gatewayed response from SMS to E-mail. By default, the US-ASCII string "Re: " is used. The string should be specified using the same character set as that specified by the `email_header_charset` option.

53.2.7 Address extraction SMS options: parse_re_N (regular expression)

For mobile origination of e-mail, the gateway profile needs to extract a destination e-mail 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 e-mail address is found or the list of regular expressions exhausted.

Note: Use of the `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 and the option `text_to_subject` has the value 0, 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 e-mail 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 e-mail 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 e-mail address, [email_header_charset](#) for the Subject: text, if any, and [email_body_charset](#) for the message body, if any.

When the option [text_to_subject](#) has the value 1, then the default value for [parse_re_0](#) is instead,

```
[ \t]*([^\( ]*)[ \t]*(.*)
```

With this default, the ability to split the SMS text between the resulting e-mail message's Subject: header line and body is no longer present. The entire SMS message is placed in the Subject: header line. This regular expression only describes how to distinguish the recipient's e-mail address from the remainder of the SMS message.

53.2.8 profile Option

The [profile](#) [SMS gateway profile option](#) specifies a 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](#) has the value 0 (zero) which is the default setting for that option.

The permitted values for this option are "GSM", "TDMA", or "CDMA". The string "ANSI-136" is treated as a synonym for TDMA; the string "IS-95" as a synonym for "CDMA".

53.2.9 route_to Option

The `route_to` SMS `gateway_profile` option specifies an IP host name to which all SMS messages targeted to the profile will be rerouted, using an e-mail address of the form:

SMS-destination-address@route-to-value

where *SMS-destination-address* is the SMS message's destination address, and the *route-to-value* is the IP host name specified with this `route_to` option. The entire content of the SMS message is sent as the content of the resulting e-mail message.

Note: Use of `parse_re` and `route_to` options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

53.2.10 select_re Option

The `select_re` SMS `gateway_profile` option specifies 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 e-mail 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.

53.2.11 smsc_default_charset Option

The `smsc_default_charset` SMS `gateway_profile` option specifies 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.

53.2.12 use_sms_priority Option

By default, priority flags in SMS messages are ignored and not sent with the e-mail messages. To have the priority flags passed with the e-mail, specify `use_sms_priority=1`. When passed with the e-mail, the mapping from SMS to e-mail is as shown in [Table of SMS Priority Flag Mappings to E-Mail](#).

Table 53.2 SMS Priority Flag Mappings to E-Mail

SMS Profile	SMS Priority Flag	Resulting E-Mail Priority: Header Line
GSM	0 (non-priority) 1, 2, 3 (priority)	No header line (implies "normal") Urgent
TDMA	0 (bulk) 1 (Normal) 2 (Urgent) 3 (Very Urgent)	Nonurgent No header line (implies Normal) Urgent Urgent
CDMA	0 (Normal) 1 (Interactive) 2 (Urgent) 3 (Emergency)	No header line (implies Normal) Urgent Urgent Urgent

Note that the e-mail Priority: header line values are Nonurgent, Normal, and Urgent.

53.2.13 use_sms_privacy Option

By default, SMS privacy indications are ignored and not sent with the e-mail messages. To have this information passed with the e-mail in a Sensitivity: header line, specify a value of 1 for the use_sms_privacy SMS gateway_profile option. When passed along in e-mail, the mapping from SMS to a Sensitivity: header line is as shown in [Table of Privacy Flag Mappings from SMS to E-Mail](#),

Table 53.3 Privacy Flag Mappings from SMS to E-Mail

SMS Privacy Flag	Resulting E-Mail 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 e-mail Sensitivity: header line are Personal, Private, and Company-confidential.

53.3 SMS smpp_relay options

There are a number of options under smpp_relay.

53.3.1 backlog Option Under smpp_relay

The backlog SMS smpp_relay option (in legacy configuration, LISTEN_BACKLOG) specifies the size of the connection backlog for inbound SMPP client connections.

53.3.2 listen_addresses Option Under smpp_relay

The listen_addresses SMS smpp_relay option (in legacy configuration, LISTEN_INTERFACE_ADDRESS) specifies the network interface for inbound SMPP client connections. If this option is not set, then listen on INADDR_ANY (that is, all addresses).

53.3.3 listen_receive_timeout Option

The listen_receive_timeout option, available under smpp_relay and smpp_server, specifies the timeout (in seconds) to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

53.3.4 listen_transmit_timeout Option

The listen_transmit_timeout option for the SMS smpp_relay and smpp_server specifies a timeout (in seconds) to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

53.3.5 make_source_addresses_unique Option

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 e-mail originator.

To disable this generating of unique SMS source addresses (for one-way SMS), specify a value of 0 (zero) for the `make_source_addresses_uniqueSMS smpp_relay` option.

53.3.6 max_conns Option Under smpp_relay

The `max_conns` [SMPP Relay option](#), `smpp_relay.max_conns`, specifies the maximum number of concurrent, inbound TCP connections to allow for this SMPP Relay instantiation. By default, a value of 7,000 connections is used. Note that this option will be ignored if it exceeds the global `sms_gateway.max_conns` setting.

53.3.7 server_host Option Under smpp_relay

The `server_host` [SMPP Relay option](#) specifies the SMPP server to which to relay SMPP client traffic. Either a hostname or IP address may be specified. Specification of this option is mandatory; there is no default value for this option.

53.3.8 server_port Option Under smpp_relay

The `server_port` [SMS smpp_relay option](#) specifies the 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.

53.3.9 server_receive_timeout Option

The `server_receive_timeout` [SMPP Relay option](#) specifies the timeout (in seconds) to allow when waiting to read data from the SMPP server. The default value is 600 seconds (10 minutes).

53.3.10 server_transmit_timeout Option

The `server_transmit_timeout` [SMPP Relay option](#) specifies the timeout (in seconds) to allow when sending data to the SMPP server. The default value is 120 seconds (2 minutes).

53.3.11 tcp_ports Option Under smpp_relay

The `tcp_ports` [SMS smpp_relay option](#) specifies the TCP port for inbound SMPP client connections. Specification of this option is mandatory; there is no default value. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.

53.4 SMS smpp_server options

There are a number of options under `smpp_server`.

53.4.1 backlog Option Under smpp_server

The backlog SMS `smpp_server` option (in legacy configuration, `LISTEN_BACKLOG`) specifies the size of the connection backlog for inbound SMPP server connections.

53.4.2 `esme_address_npi` Option

By default, when binding as a receiver in response to an `OUTBIND` request, the SMPP server will specify an ESME Numeric Plan Indicator (NPI) value of zero indicating an unknown NPI. 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 case-insensitive text strings shown in [Supported ESME NPI types](#)

Table 53.4 Supported ESME NPI types

String	Hexadecimal Value
data	0x03
default	0x00
e.163	0x01
e.164	0x01
e.212	0x06
ermes	0x0a
f.69	0x04
internet	0x0e
ip	0x0e
isdn	0x01
land-mobile	0x06
national	0x08
private	0x09
telex	0x04
unknown	0x00
wap	0x12
x.121	0x03

This option is ignored unless the `server_port` is also specified. See the description of the [server_port](#) option for further information.

53.4.3 `esme_address_range` Option

The ESME address range to present when binding as a receiver in response to an `outbind` request. By default, an empty string is used. For some SMSCs, an address range of “[:alnum:]*” may be appropriate. Any string specified must not exceed a length of 40 bytes.

53.4.4 `esme_address_ton` Option

By default, when binding as a receiver in response to an OUTBIND request, the SMPP server will specify an ESME Type of Number (TON) value of zero indicating an unknown TON. Values for this option may be specified in one of three ways,

- A decimal value (for example, 1).
- A hexadecimal value prefixed by “0x” (for example, 0x01).
- One of the case-insensitive text strings shown in [Table of supported ESME TON types](#)

Table 53.5 Supported ESME TON types

String	Hexadecimal Value
abbreviated	0x06
alphanumeric	0x05
default	0x00
international	0x01
national	0x02
network-specific	0x03
subscriber	0x04
unknown	0x00

This option is ignored unless the `smpp_server.server_port` option is also specified. See the description of the [server_port](#) option for further information.

53.4.5 esme_password Option

When binding as a receiver in response to an OUTBIND request, the SMPP server must specify an ESME password. By default, an empty string is used for the password. Use this option to specify a non-empty string. As with all ESME passwords, the maximum length of this string is 8 bytes.

53.4.6 esme_system_id Option

ESME system id to present when binding as a receiver in response to an OUTBIND request. By default, an empty string is used. Use this option to specify a string at most 15 bytes long.

53.4.7 esme_system_type Option

The `esme_system_type smpp_server` option specifies the ESME system type to present when binding as a receiver in response to an OUTBIND request. By default, an empty string is used. Use this option to specify a string at most 12 bytes long.

53.4.8 listen_addresses Option Under smpp_server

The `listen_addresses SMS smpp_server` option (in legacy configuration, `LISTEN_INTERFACE_ADDRESS`) specifies the network interface for inbound SMPP server connections. If this option is not set, then listen on `INADDR_ANY` (that is, all addresses).

53.4.9 listen_receive_timeout Option

The `listen_receive_timeout` option, available under `smpp_relay` and `smpp_server`, specifies the timeout (in seconds) to allow when waiting to read data from an SMPP client. The default value is 600 seconds (10 minutes).

53.4.10 listen_transmit_timeout Option

The `listen_transmit_timeout` option for the SMS `smpp_relay` and `smpp_server` specifies a timeout (in seconds) to allow when sending data to an SMPP client. The default value is 120 seconds (2 minutes).

53.4.11 max_conns Option Under smpp_server

The `max_conns` [SMPP Server option](#), `smpp_server.max_conns`, specifies the maximum number of concurrent, inbound TCP connections to allow for this SMPP Server instantiation. By default, a value of 7,000 connections is used. Note that this option will be ignored if it exceeds the global `sms_gateway.max_conns` setting.

53.4.12 server_host Option Under smpp_server

The `server_host` [SMPP Server option](#) specifies the remote host to bind back to in response to an OUTBIND request. This option is ignored unless the `server_port` is also specified. See the description of the [server_port](#) option for further information.

53.4.13 server_port Option Under smpp_server

Some SMSCs incorrectly implement the OUTBIND operation by sending an OUTBIND PDU and then closing the TCP connection. As per the SMPP specification, the SMSC is expected to leave the connection open, allowing the receiver of the OUTBIND to then bind back as receiver over the same connection. Use the `server_port` SMPP Server option to interoperate with such SMSCs by specifying the TCP port to bind back to when an OUTBIND request is received. If the [server_host](#) SMPP Server option is not specified, then the source IP address associated with the current TCP connection is used.

See also the [esme_address_npi](#), [esme_address_range](#), [esme_address_ton](#), [esme_password](#), [esme_system_id](#), and [esme_system_type](#) options.

53.4.14 system_id Option

An optional SMPP `system_id` string of zero to fifteen (0-15) US-ASCII characters may be specified. This string will then be returned as the SMPP server's system id in bind responses sent to SMPP clients. By default, an empty system id string is returned.

53.4.15 tcp_ports Option Under smpp_server

The `tcp_ports` SMS `smpp_server` option specifies the TCP port for inbound SMPP server connections. Specification of this option is mandatory; there is no default value. Note also that there is no Internet Assigned Numbers Authority (IANA) assignment for this service.



Chapter 54 Message capture

54.1 MESSAGE-SAVE-COPY mapping table	54-3
54.1.1 MESSAGE-SAVE-COPY mapping table format and examples	54-4
54.1.2 Message replay of captured message copies	54-5
54.2 Capture triggered via LDAP attributes	54-6
54.3 Capturing messages via Sieve scripts	54-6
54.4 Format of captured message copies	54-7
54.5 Archiving messages	54-17
54.5.1 Choosing which messages to archive	54-17
54.5.2 Message identifier generation	54-18
54.5.3 AXS:One archive integration	54-20

Message capture may be desired for purposes including: archiving, lawful interception, covert or administrative monitoring, or message replay (disaster recovery). The MTA has a number of facilities that can be used to "capture" messages, taking the message "outside" the normal message processing flow; this can be useful for tasks such as: monitoring (without a user's knowledge) the messages sent and received by the user such as for lawful interception purposes, or for making copies of messages passing through the MTA, possibly for archival purposes or to allow for possible future "message replay" as part of a disaster recovery strategy.

Note that the facilities discussed here are fundamentally different in spirit (as well as in details) from techniques such as automatically forwarding messages to an additional address---forwarding techniques that have the potential, as part of normal e-mail processing in cases of delivery problems for the "forwarded" message copy, to result in exposure of the fact of message "forwarding" or "copying" to end users, or which may, as part of normal e-mail processing in cases of group or alias expansion problems, prevent end users from being notified of certain sorts of recipient address problems even for the recipient(s) the end user did knowingly address. (That is, techniques such as adding LDAP attributes `mailDeliveryOption` with value `forward` and `mailForwardingAddress` to user LDAP entries, or use of a [FORWARD mapping table](#), or use of "redirect" Sieve actions, are not discussed here.) Rather, the techniques discussed here are those that have as a fundamental aspect the goal of separate handling of the "captured" message copies, techniques where the fact/process of copying is invisible to the end users. *Message capture is distinct from simple message forwarding.*

The main techniques that the MTA provides for interception/covert/archival "capture" of messages are:

- The [MESSAGE-SAVE-COPY mapping table](#), used to copy message files from the MTA's disk queue area. This facility was originally designed for, and is especially well-suited for, making short-term copies of outbound messages for possible "message replay" (re-sending) in case of loss of messages on the destination host(s).
- The LDAP "capture" [user attribute](#), used to capture copies of all messages sent or received by the user (by generating encapsulated copies of the messages and directing them to a specified capturer address), discussed in [Capture triggered via LDAP attributes](#). This facility was originally designed for, and is especially well-suited for, monitoring of individual users' message traffic (*e.g.*, for legal purposes or administrative monitoring purposes). Or, by capturing such messages in Microsoft[®] Exchange "envelope journaling" format, the captured

message copies may be convenient for archiving purposes; see the (new in MS 7.0u4) [capture_format_default](#) MTA option, or (new in MS 8.0) use an LDAP tag `;format-journal-header` on the LDAP attributes named by the MTA options [ldap_capture](#) and (also new in MS 8.0) [ldap_domain_attr_capture](#).

- (New in MS 8.0) The [LDAP "capture" domain attribute](#) (a domain-level analogue of the LDAP "capture" user attribute), used to capture copies of all messages sent or received by users in that domain, discussed in [Capture triggered via LDAP attributes](#).
- (New in MS 6.2) The [CAPTURE named parameter for aliases and mailing lists](#) defined in the MTA [alias file](#) operates similarly to the [LDAP "capture" user attribute](#). For the syntax of the CAPTURE named parameter for simple aliases and for groups or mailing lists, see [Alias file format](#) and [Alias file named parameters](#). Nowadays MTA alias file definitions are less commonly used than LDAP provisioning of users and lists—but the CAPTURE named parameter is provided as an alternative that may be convenient for sites that do make more use of the MTA alias file. In Unified Configuration, the [alias_capture](#) alias option is the equivalent of the alias file named parameter CAPTURE.
- (New in MS 7.2-0.01) The [JOURNAL named parameter for aliases and mailing lists](#) defined in the MTA [alias file](#) operates similarly to the CAPTURE named parameter, but generates a Microsoft Exchange "envelope journaling" format message; this format may be especially useful for archiving purposes. In Unified Configuration, the [alias_journal](#) alias option is the equivalent.
- The MTA's address access mapping tables (see [Address access mapping table flags](#)) can be configured to trigger message capture via the `$M` flag; and new in MS 7.0.5, such captured messages can optionally be generated in Microsoft Exchange "envelope journaling" format, configured via the `$+L` flag. (Address access mapping table triggered capture is a less commonly used feature: although it allows for greater discrimination than a user LDAP "capture" attribute, since for instance a mapping entry might be configured to capture only messages from one specific sender to another specific recipient, it is less discriminating than use of a [Sieve "capture" action](#). So unless configuration in an access mapping table is particularly convenient, more commonly some other technique such as Sieve "capture" would be employed.)
- The [Sieve "capture" action](#), used to capture copies of messages meeting any Sieve-specifiable criteria, and directing encapsulated copies of the messages to a specified capturer address, discussed in Section 26.3. Because of Sieve flexibility, this is especially well-suited for capturing only specific categories of messages: messages meeting some rather specific (and Sieve specifiable) criteria. (New in JES MS 6.3, messages captured via a Sieve filter may optionally be sent without MIME encapsulation, but with an override of the original envelope From address. This new option for capture may be of special interest for archiving purposes, when the simpler, unencapsulated message form may be more convenient. Or yet another option, new in MS 7.0 update 2, is that such capture messages may be generated in Microsoft Exchange's "envelope journaling" format: a multipart MIME message where the first part contains semi-structured envelope information and the second part contains the actual original message. "envelope journaling" format may be more convenient for archiving purposes.)
- (New in JES 5 a.k.a MS 6.3.) Integration with the AXS:One archive facility, used to generate message copies that will be archived by AXS:One, discussed in Section 26.5. This is primarily suitable for compliance archiving.

Note that with any of the techniques discussed below, issues of use of "captured" message copies, and potentially issues of correlation (and elimination of "duplicate" copies of the "same"

message captured at different stages of processing) may arise; it is the responsibility of sites to devise strategies appropriate for their goals.

Note also that any "capture" of users' messages may, indeed is likely to, have legal ramifications. Sites are cautioned to **obtain legal advice** before beginning any use of message capture techniques.

54.1 MESSAGE-SAVE-COPY mapping table

The MESSAGE-SAVE-COPY mapping table provides a way to tell the MTA to copy (rename) message files from its disk queue area when dequeuing messages. The resulting files are in the MTA's disk queue area (proprietary) format. When copied (renamed) to some area outside the MTA's normal disk queue area, these copies of the MTA's message files are no longer subject to normal processing by the MTA. But if subsequently renamed back into the MTA's disk queue area, the message files are perfectly suited to being "picked up" by the MTA, and re-sent just as the original message files were sent.

The original motivation for this facility was as a means to capture copies of messages outbound from the MTA to be retained (for some period) for purposes of potential future "message replay", that is, it was intended to be used to resend messages in case of disaster on the original receiving host(s). This facility can potentially be used, however, for other purposes such as monitoring (capturing) of messages sent by particular users, or as a way to obtain message copies for long-term archiving purposes.

When using this facility for purposes of monitoring particular users' e-mail messages, keep in mind that while capturing of messages sent *from* a particular user is straightforward enough (the MESSAGE-SAVE-COPY mapping table probes include the envelope From address as part of the probe field), the capturing of messages sent *to* a particular user is less straightforward with this facility, as it normally distinguishes only between classes of destination addresses---that is, it distinguishes on a destination channel basis (as the MESSAGE-SAVE-COPY mapping table probes include the destination channel, but not the envelope To recipient(s)). Thus if it is desired to use this facility to capture specifically those messages to a particular user or users, it may be necessary to either capture additional messages (to other recipients) as well and then in some site-supplied subsequent processing discard the undesired messages, or else to perform some additional, more complex, configuration of the MTA to allow the MESSAGE-SAVE-COPY mapping table to distinguish which messages are to the recipient(s) in question.

When using this facility to obtain message copies that will then be further processed, *e.g.*, delivered into an archiving system, note that the copies created are in MTA message file format. All access to such message files should be done via the MTA SDK. (Essentially, a channel program should be written to process the messages.) Accessing the message files only through the MTA SDK insulates applications from potential future changes in the MTA's message file structure (particularly, changes and additions to the message envelope portion of the message file).

The choice of when the MESSAGE-SAVE-COPY mapping table should be applied -- for which destination channel(s), and in a multi-host e-mail environment, on which hosts---should be carefully considered in relation to the fundamental goals of the message capture. And typically some thought needs to be given to issues of message "split up" -- the cases where a multi-recipient message may bifurcate into separate "copies" due to different recipients needing different types of handling -- in relation to the message capture goals. If the goal is simply to capture "all messages going out a particular channel" (the case for which this facility was designed), achieving that goal is simple. But otherwise, other questions arise. Is the goal to capture each such message copy? Will there be a desire to "correlate" or "consolidate" the

cases where separate eventual message copies all correspond to a single, originally submitted message? In a multi-host environment, is it desired to capture merely those messages passing through a particular host? Or do messages potentially need to be captured on multiple hosts and if so, does there need to be some "correlation" or "consolidation" of the message copies captured on different hosts (some of which copies may consist of the "same" message, at a different point in its processing life-cycle)? As regards message "split-up", a most basic example would be the case of a multi-recipient message where the recipients are destined out different channels. But there are also many other sorts of message processing that imply separate handling via separate message files, such as local recipients with different [conversion tags](#), or mailing list recipients *vs.* directly addressed recipients, *etc.* Even on a single system, the potential use of any "intermediate" channels such as the [conversion](#) or [reprocess](#) channels should be considered in relation to timing of the MESSAGE-SAVE-COPY operation. And when operating in a multi-host environment, the timing and message bifurcation issues tend to become more complex.

54.1.1 MESSAGE-SAVE-COPY mapping table format and examples

The format of a MESSAGE-SAVE-COPY mapping table entry is, by default:

```
out-channel | return-address | D | orig-file-path  $Yresult-file-path
```

where *out-channel* is the destination channel out which the message is being dequeued, *return-address* is the envelope From address, and *orig-file-path* and *result-file-path* are full file path specifications. Note that the template (right hand side) of the mapping table entry must include one of the flags *\$Y*, *\$y*, *\$T*, or *\$t* in order for the rename to be attempted.

If the new-in-6.3 [message_save_copy_flags](#) MTA option has all of its bits set (corresponding to a value of 7), then the probe format instead becomes

```
transport-info | app-info | source-channel | conv-tags | out-channel | return-address | D | orig-file-path
```

where the *transport-info* and *app-info* are as usually defined (*transport-info* in particular corresponding to the fields seen in a [PORT_ACCESS](#) probe) -- see for instance their discussion in the discussion of the [MAIL_ACCESS](#) mapping table or in the discussion of the [log_connection](#) MTA option; where *source-channel* is the original source channel; and where *conv-tags* consists of any current [conversion tags](#).

As a rename operation is used to rename the original file to the result (copied) file, the result file specification must be on the same disk as the original file path, and the path must be writable by the MTA. However, normally an area under the IMTA_QUEUE area should not be used as the result location, as that is the MTA's area for message files that it expects to be eligible for MTA automatic processing. Thus normally an area on the same disk, and owned by the MTA, but outside the actual IMTA_QUEUE area itself, should be used as the place to which to copy (rename) the message files.

As of the 8.0 release, MESSAGE-SAVE-COPY provides a means of copying message files instead of, or in addition to, renaming them. If *\$G* is specified, then a file name is read from the mapping result and the current message file is copied there. If both *\$G* and *\$Y* are specified, then the file is both copied and renamed; in this case the mapping result must be of the form:

```
$Y$Gcopy-file-name | rename-file-name
```

If \$Q is specified in addition to \$Y, then an attempt will be made to tell the [Job Controller](#) to process the message file in its new location. \$Q is intended to be used when a message file is moved from one queue to another.

Also new in 8.0 is the \$S flag. \$S can be used in a MESSAGE-SAVE-COPY mapping to say that the message file has been renamed or otherwise processed by the mapping template and the file should simply be closed, not deleted or otherwise modified. \$S is only effective if \$Y is not specified.

As an example, to capture a copy of each message file being dequeued out to the Internet (out the [tcp_local channel](#)), a mapping table as follows might be used:

MESSAGE-SAVE-COPY

```
tcp_local | * | D | /opt/SUNWmsgsr/data/queue/tcp_local/%%%/* \
    $Y/opt/SUNWmsgsr/msg-save/tcp_local/$1$2$3/$4
```

Note how in the above example the assumption is that the [subdirs](#) channel option is in use on the tcp_local channel (so that message files are stored in subdirectories under the tcp_local channel's disk queue area), and how the subdirectory structure is preserved due to the same values being substituted back in via the template.

As another example, to capture a copy of message file being dequeued out to the Message Store via the [ims-ms channel](#), a mapping table such as the following might be used:

MESSAGE-SAVE-COPY

```
ims-ms | * | D | IMTA_QUEUE:ims-ms/%%%/* \
    $Y/opt/SUNWmsgsr/msg-save/ims-ms/$1$2$3/$4
```

To capture all messages from a particular user is straightforward via the MESSAGE-SAVE-COPY mapping table, by specifying that user's address as the return-address in the probe.

However, capturing all messages to a particular user but only to that particular user via this facility would require a more complex configuration of the rest of the MTA: prior to JES MS 6.3, such a task would typically require use of a special delivery channel -- see [Additional ims-ms channels](#) -- or in JES MS 6.3 or later, an alternative approach for capturing messages to some special user(s) would be to configure the user(s) in question with some special [mailConversionTag](#) value, set the [message_save_copy_flags](#) MTA option to 4 (or some value including 4 in the bit mask) so that probes to MESSAGE-SAVE-COPY include [conversion tags](#), and then use conversion tag sensitive entries in the MESSAGE-SAVE-COPY mapping table.

54.1.2 Message replay of captured message copies

Message files captured via the [MESSAGE-SAVE-COPY mapping table](#), if moved back into appropriate MTA channel queue area(s), are perfectly suited to being "replayed"---redelivered.

When performing "message replay" of message files captured using the MESSAGE-SAVE-COPY mapping table, note that the Job Controller option `rebuild_in_order` may be of interest, especially if combined with use of the `imsimta qm utility's stop channel-name` command. Or, rather than setting `rebuild_in_order` "permanently" in the Job Controller

configuration file, [imsimta cache -change](#) commands can, as of JES MS 6.2, be used to enable and disable "in order" rebuild for an already running Job Controller. Thus a command sequence to "replay" captured copies of messages previously sent out the `channel-name` channel might look something like:

```
# imsimta qm stopchannel-name
# imsimta qm cache -change -global -inorder_rebuild
# mvsaved-message-files/opt/SUNWmsgsr/data/queue/channel-name
# imsimta cache -synchronize
# imsimta qm startchannel-name
# imsimta qm cache -change -global -noinorder_rebuild
```

54.2 Capture triggered via LDAP attributes

The LDAP capture attribute (the exact attribute name is site-chosen, and specified via the [ldap_capture](#) MTA option) provides a way to tell the MTA to "capture" a copy of each message sent *to* or *from* a user who has the attribute present. The capture copy will be sent to the value (the address) specified in the LDAP capture attribute. Normally, the LDAP capture attribute itself should be configured in the LDAP directory as an [attribute that the user can neither set nor even see](#) themselves; that is to preserve the covert nature of the LDAP capture attribute.

Multiple capture attributes may apply to a particular user, or particular message copy (due either to multiple attributes on one user, or to attributes on both sender and recipient(s)).

New in MS 8.0, the MTA also supports enabling capture at the domain level; see the [ldap_domain_attr_capture](#) MTA option.

See [Format of captured message copies](#) for a discussion of the format of captured message copies.

Note that (LDAP attribute triggered) capture of messages that a user *sends* is triggered during [address reversal](#), and hence in order to capture the messages that the user sends, it is critical to be performing address reversal, and in particular properly configured address reversal. See [Intended side effects of LDAP address reversal](#).

(In the interests of symmetry and completeness, it could be noted that (LDAP attribute triggered) capture of messages *to* a user is triggered during [LDAP alias expansion](#) for the user ([alias_urlN](#) lookups), so for capture of messages to a user it is critical that such LDAP alias lookups be configured as normal. However, LDAP alias lookups are such a fundamental part of normal MTA operation, that unless a site has intentionally modified their configuration in abnormal ways, it would be very unusual for this to be a concern. This is in contrast to [address reversal](#) which, though strongly recommended nowadays, may still be omitted from older configurations at some sites.)

54.3 Capturing messages via Sieve scripts

The MTA has a private Sieve extension action, "[capture](#)", that takes an argument specifying a destination address to which to send an encapsulated copy of the original message. "capture" is supported only in [system Sieves](#): that is, channel Sieves or the MTA [systemfilter](#) (in legacy configuration, the `imta.filter` file). By default, this causes generation of a new message to the capturer in almost exactly the same [DSN format](#) (see [Format of captured](#)

[message copies](#)) as results from use of the `ldap_capture` attribute (discussed in [Capture triggered via LDAP attribute](#)).

New in JES MS 6.3 are the optional, nonpositional parameters `:dsn` and `:message`. The default is `:dsn`, which corresponds to the only behavior previously available, that of captured copies being in encapsulated format as with `ldap_capture` attribute use. However, if the new-in-6.3 `:message` parameter is specified, then the "capture" message copy may instead be generated without MIME encapsulation (though the envelope From address of the new message copy will be set to that of the ["owner" of the Sieve filter](#)).

New in MS 7.0 update 2 is the optional parameter `:journal`, which is an alternative to `:message` or `:dsn`. This new `:journal` parameter causes the Sieve "capture" action to produce Microsoft Exchange's "envelope journaling" format. This format consists of a multipart MIME message where the first part contains envelope information in a semi-structured format and the second part is the actual message.

Because the "capture" action can, like any other Sieve action, be coded into use in a Sieve script doing complicated filtering of messages, it is especially well-suited to cases where it is desired to "capture" only particular sorts of messages (*e.g.*, those containing particular header lines, particular combinations of senders and recipients, particular sorts of message contents, *etc.*). For some examples, see [Example Sieve external lists with properties](#).

However, Sieve filter based "capture", especially with the new-in-6.3 `:message` argument or the new-in-7.0u2 `:journal` argument, may also be a useful part of an archiving approach. See [Format of captured message copies](#) for a discussion of the possible formats for captured message copies.

54.4 Format of captured message copies

"Captured" message copies by default are in the form of [Delivery Status Notifications](#) (see [RFC 1892](#)). So for instance, if an original message has the form shown in [Original message to be captured, as submitted](#) at the point when the MTA applies capture (for instance capture performed by the SMTP server when the message is first submitted)---and where, for comparison, that original message by the time it is delivered to a user mailbox might have a form as shown in [Original message as delivered](#):

Original message to be captured, as submitted

```
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>

an original line of text
```

Original message as delivered

```
Return-path: <user1@domain.com>
Received: from [10.1.110.115] ([10.1.110.115])
  by host.domain.com (Sun Java(tm) System Messaging Server 7.3-11.01 64bit
  (built Sep 1 2009)) with ESMTTP id <0KRH00A3QAO42T10@host.domain.com>
```

Format of captured message copies

```
for user2@domain.com; Tue, 13 Oct 2009 17:28:52 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>
Original-recipient: rfc822;user2@domain.com
```

an original line of text

then a captured message copy (the copy generated and sent to the value of an LDAP attribute named by the `ldap_domain_attr_capture` MTA option or the `ldap_capture` MTA option) would have the form shown in [Captured message copy \(default DSN format\)](#). Similarly, a captured message copy generated and sent to the address named in an unadorned "capture" Sieve action would have almost exactly that same form, with only a minor text difference as discussed at [\(11\)](#).

Captured message copy (default DSN format)

```
Return-path: <> (1)
Received: from process-daemon.host.domain.com by host.domain.com (Sun Java(tm)
  System Messaging Server 7.3-11.01 64bit (built Sep 1 2009)) id
  <01NEVLK3B0VK00170V@host.domain.com> for subpoenal-on-user1@domain.com;
  Tue, 13 Oct 2009 17:28:14 -0700 (PDT)
Received: from host.domain.com (Sun Java(tm) System Messaging Server
  7.3-11.01 64bit (built Sep 1 2009)) id <01NEVLJLOOF600156Q@host.domain.com>;
  Tue, 13 Oct 2009 17:28:11 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:28:11 -0700 (PDT)
From: Internet Mail Delivery <postmaster@host.domain.com> (2)
Subject: Message Capture Copy (3)
To: subpoenal-on-user1@domain.com (4)
Message-id: <0IH400G08EULG800@ketu.west.sun.com>
MIME-version: 1.0
Original-recipient: rfc822;subpoenal-on-user1@domain.com
Content-type: multipart/report; report-type=delivery-status;
  boundary="Boundary_(ID_nChZX4aV2kgbzSzzoDGCvw)"

--Boundary_(ID_nChZX4aV2kgbzSzzoDGCvw)
Content-type: text/plain; charset=us-ascii (5)
Content-language: en-US

This report relates to a message you sent with the following header fields:

  Message-id: <0IH400G04EU3G800@host.domain.com> (6)
  Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT)
  From: user1@domain.com
  To: user2@domain.com
  Subject: test message

Attached message captured in accordance with site policy. (7)
```

```

--Boundary_(ID_nChZX4aV2kgbzSzzoDGCvw)
Content-type: message/delivery-status

Reporting-MTA: dns;host.domain.com (tcp_intranet-daemon)      (8)
Arrival-date: Tue, 13 Oct 2009 17:28:02 -0700 (PDT)          (9)

Original-recipient: rfc822;user2@domain.com                  (10)
Final-recipient: rfc822;user2@domain.com
Action: capture
Status: 2.0.0 (Copy requested by capture attribute)          (11)

--Boundary_(ID_nChZX4aV2kgbzSzzoDGCvw)
Content-type: message/rfc822

Return-path: <user1@domain.com>                               (12)
Received: from [10.1.110.115] ([10.1.110.115])               (13)
  by host.domain.com (Sun Java(tm) System Messaging Server 7.3-11.01 64bit
  (built Sep  1 2009)) with ESMTTP id <0KRH00A3QAO42T10@host.domain.com>;
  Tue, 13 Oct 2009 17:28:11 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT)                 (14)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>

an original line of text

--Boundary_(ID_nChZX4aV2kgbzSzzoDGCvw)--

```

In the captured message copy, note the following items of interest:

1. As with any DSN, note that the envelope From address on this DSN is empty. (Note that the addition of a Return-path: header line only occurs at final message delivery time; if looking at the capture message copy earlier, such as while it is transitting the MTA, this header line would not be expected to be present.)
2. Captured message copies are encapsulated, with the new, encapsulating message having the form of a notification message: an empty envelope From and a From: header value set to the postmaster address. (By default, the host's [postmaster address](#) is used, though an override postmaster address may be used if domain-specific postmasters are configured --- see the [ldap_domain_attr_report_address](#) MTA option --- or if use of an override postmaster address has been set by the [FROM_ACCESS](#) mapping table.)
3. The Subject: value for captured message copies defaults to the string "Message Capture Copy", or may be set to an alternate string (then used for all DSNs in the relevant language choice) using the [SUBJECT](#) option in the [return_option.opt](#) file, or may be modified on a per-type-of-DSN basis using the \$T flag in the appropriate [NOTIFICATION_LANGUAGE](#) mapping table entries.
4. The message capture copies are sent to the address specified as the value of the attribute named by the [ldap_capture](#) or (new in MS 8.0) the [ldap_domain_attr_capture](#) MTA option; as for instance in this example the attribute is assumed to have the value `subpoenal-on-user1@domain.com`. (Or in the case of [Sieve "capture" actions](#), the message capture copies are sent to the address specified in the "capture" action.)

5. The [return_prefix.txt](#) file is used to construct the the first part of DSN messages: specifically, the MIME header lines, plus (typically, and in particular in this example) the introductory text line and insertion of a sample of the original message headers. Normally, the [NOTIFICATION_LANGUAGE mapping table](#) is configured to choose an appropriate, language-specific such set of MIME header lines and introductory text.
6. When the [return_prefix.txt](#) file uses the %H substitution, exactly which header lines from the original message that will cause to be included in this first part normally is controlled, for all DSNs of all types in all languages, by the `return_header.opt` file. (New in MS 7.0 update 2, use of language-specific variants of `return_header.opt` may be configured.) That is, the use of %H substitution in `return_prefix.txt` causes inclusion of headers, with `return_header.opt` controlling exactly which headers.
7. The appropriate, language-specific [return_capture.txt](#) file specifies the text added here at the bottom of the first part of the DSN in the case of message capture copies.
8. The host and channel that are generating the DSN (the captured message copy in this case) are reported.
9. As of JES MS 7.0, the arrival date of when the message "arrived" at the reporting MTA is reported on an "Arrival-date:" line. Note that as this represents the time when the MTA *began* processing this message, it will tend to be a time slightly prior to the time of the Received: header line that the MTA added on this "capture" message copy (though this time is of course after the time at which the original message was composed).
10. For each current recipient of the message, a set of fields is output, reporting on that recipient. Normally this set consists of an "Original-recipient:" field, a "Final-recipient:" field (note that the [useintermediate](#) and [suppressfinal](#) channel options alter what is actually reported here), an "Action:" field (which is of course "capture" for capture copies), and a "Status:" field (for capture copies due to [ldap_domain_attr_capture](#) or [ldap_capture](#), this field will contain a string along the lines of "Copy requested by capture attribute"; or for capture copies due to Sieve "capture", "Copy requested by capture filter").
11. The one difference between a captured copy due to use of the [ldap_domain_attr_capture](#) or [ldap_capture](#) attribute *vs.* use of a [Sieve unadorned "capture" action](#) is whether the text reported on the Status: line says "Copy requested by capture attribute" (or one of the new in MS 8.0 variants corresponding to tagged LDAP attribute value "Copy requested by journal attribute", "Copy requested by capture header attribute", "Copy requested by journal header attribute") *vs.* "Copy requested by capture filter".
12. The envelope From of the original message is reported on the Return-path: header line of the included original message.
13. For the captured message copy, a Received: header line closely corresponding to the Received: header line that was constructed for the original message during the processing at the time that the original message was captured, is constructed and reported here.
14. The entire header (all header lines) of the original message is included, as it existed at the point of capture.

Note that as usual with DSNs, some customization of the text intended to be human-readable is possible. The text value on the Subject: header line may be modified (for all DSNs in the relevant language choice) via the [SUBJECT option in the return_option.opt](#) file, or may be modified on a per-type-of-DSN basis using the \$T flag in the appropriate

[NOTIFICATION_LANGUAGE mapping table](#) entries. The first part of the DSN (the TEXT/PLAIN part), may be localized or site modified via the [return_prefix.txt file and the return_capture.txt file](#) corresponding to the relevant language choice, as well as the general [return_header.opt file](#). See [DSN language and customization](#) for further details.

When using an [ldap_capture](#) or (new in MS 8.0) an [ldap_domain_attr_capture](#) named LDAP attribute, the regular, DSN form of captured message copy, as shown in [Captured message copy \(default DSN format\)](#) (or with the slight variations configurable via DSN configuration as discussed there) is (prior to MS 8.0) always generated. (New in MS 8.0, the LDAP attributes named by the [ldap_capture](#) and [ldap_domain_attr_capture](#) MTA options supported tagged values, where the tags select the format to be generated. So as of MS 8.0, LDAP attribute triggered capture can also cause generation of "journal" format, or of modified DSN or "journal" format containing only headers of the original message -- which may be useful for administrative monitoring that preserves message content privacy.) That DSN format is also (essentially) the format generated by default when using the Sieve "capture" action. But when using the Sieve "capture" action, two other alternate formats may be generated.

Requesting "capture :message" results in a format such as shown in [Captured :message message copy](#), where the capture copy is not much changed from the original message copy, other than the original envelope From address being overridden for the capture copy to be that of the relevant system Sieve's "owner" (normally the postmaster address), and differences in Received: header line(s) corresponding to the new routing of the capture message copy. Although such a "capture :message" copy does not have as much information regarding the original message as would a plain "capture" copy or even a "capture :journal" copy (as discussed in [Captured journal 2003 format message copy](#)) --- specifically, this form of capture does not have fields in which to record the original message copy's envelope From and envelope To values --- for some purposes it may provide sufficient information and its "simple" structure may be convenient. However, because the fact that this is a capture message copy is not called out obviously in the capture message copy's header, the recipient of such capture copies will *need to be alert* to the reason why he or she has received the capture copy! Hence, typically the mailbox receiving such capture copies should be either: (a) a special mailbox dedicated to receiving such capture copies, or (b) the mailbox of a sophisticated e-mail user who knows to expect capture copies, and knows to look carefully at all messages received so as to detect which messages are being received due to capturing and handle such received messages appropriately. (For instance, if a dedicated mailbox is not available, consider directing capture copies to a distinguished subaddress of the mailbox, combined with a [Sieve filter that detects the presence of the subaddress](#) to cause special handling, such as delivery into a special folder.)

Captured :message message copy

```
Return-path: <postmaster@host.domain.com> (1)
Received: from host.domain.com (host.domain.com [10.1.110.114]) by (2)
  host.domain.com (Sun Java(tm) System Messaging Server 7.3-11.01 64bit
  (built Sep 1 2009)) id <01NEVMSAJ6K00015BG@host.domain.com>
  (original mail from user1@domain.com)
  for subpoena1-on-user1@domain.com; Tue, 13 Oct 2009 17:28:55 -0700 (PDT)
Received: from [10.1.110.115] ([10.1.110.115]) (3)
  by host.domain.com (Sun Java(tm) System Messaging Server 7.3-11.01 64bit
  (built
  t Sep 1 2009)) with ESMTPE id <0KRH00A3QAO42T10@host.domain.com>
  for subpoena1-on-user1@domain.com; Tue, 13 Oct 2009 17:28:52 -0700 (PDT)
```

```
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT) (4)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>
Original-recipient: rfc822;user2@domain.com

an original line of text
```

In the captured `:message` format message copy, note the following items of interest:

1. The envelope From address for a "capture `:message`" copy is the [postmaster address](#); more specifically, it is the [owner](#) of whichever system Sieve performed the "capture `:message`" action, and system Sieves are all owned by the postmaster.
2. This is the Received: header line constructed during the enqueue from the [reprocess channel](#) to the delivery channel. (Unlike the other capture formats, which are [notification messages](#) processed through the [process channel](#), the processing of a "capture `:message`" copy is more similar to the effect of a [Sieve "redirect" action](#) and in particular, (prior to MS 8.0) is handled through the [reprocess channel](#). As of MS 8.0, the process channel, rather than the reprocess channel, handles these messages too.)
3. This is the Received: header line constructed by the SMTP server enqueueing this capture message copy to the reprocess channel. Note that this Received: header line in the capture message copy is very similar to the Received: header that the SMTP server added to the original message; the difference is in the respective ["for recipient-address" clauses](#), as the capture message copy shows the recipient address for this capture message copy (if there is only one capture message recipient).
4. From here on, the copy consists of the original message (all its original header lines); in particular, the header From: is still that of the original message. Note that when capturing a message that had been relayed, the captured message copy would also contain more Received: header lines than shown here, if the original message (as in a relayed message) itself contained Received: header lines; but this example corresponds to an original message whose initial submission was to this MTA, with the capture being triggered during the processing of that initial submission.

Alternatively, again with the original message shown in [Original message to be captured, as submitted](#), a basic (2003 format) "capture `:journal`" (or an LDAP attribute triggered capture due to an LDAP attribute named by [ldap_capture](#) or [ldap_domain_attr_capture](#) having a value tagged by `;format=journal`) message copy would have the form shown in [Captured journal 2003 format message copy](#):

Captured journal 2003 format message copy

```
Return-path: <> (1)
Received: from process-daemon.host.domain.com by host.domain.com (Sun Java(tm) (2)
System Messaging Server 7.3-11.01 64bit (built Sep 1 2009)) id
<01NEVSAYXRVK0017PR@host.domain.com> for subpoena1-on-user1@domain.com;
Tue, 13 Oct 2009 17:28:25 -0700 (PDT)
Received: from host.domain.com (Sun Java(tm) System Messaging Server (3)
7.3-11.01 64bit (built Sep 1 2009)) id <01NEVSA2342U0014CT@host.domain.com>;
Tue, 13 Oct 2009 17:28:13 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:28:13 -0700 (PDT)
From: Internet Mail Delivery <postmaster@host.domain.com> (4)
```

```

Subject: Message Journal Copy (5)
To: subpoena1-on-user1@domain.com (6)
Message-id: <0KR2001I49JPZ582@host.domain.com>
MIME-version: 1.0
Original-recipient: rfc822;subpoena1-on-user1@domain.com
X-MS-Journal-Report: (7)
Content-type: multipart/mixed; boundary="Boundary_(ID_xWvHuuTuAMSGf6g0CDWjg)" (8)

--Boundary_(ID_xWvHuuTuAMSGf6g0CDWjg)
Content-type: text/plain (9)

Sender: <user1@domain.com> (10)
Message-ID: <0IH400G04EU3G800@host.domain.com> (11)
Recipients:
  <user2@domain.com> (12)

--Boundary_(ID_xWvHuuTuAMSGf6g0CDWjg)
Content-type: message/rfc822 (13)

Return-path: <user1@domain.com> (14)
Received: from [10.1.110.115] ([10.1.110.115]) (15)
  by host.domain.com (Sun Java(tm) System Messaging Server 7.3-11.01 64bit
  (built Sep 1 2009)) with ESMTP id <0KRH00A3QAO42T10@host.domain.com>;
  Tue, 13 Oct 2009 17:28:11 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT) (16)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>

an original line of text

--Boundary_(ID_xWvHuuTuAMSGf6g0CDWjg)--

```

In the captured journal format message copy, note the following items of interest:

1. The journal format is a [notification message](#): it has an empty envelope From, as reported here on the Return-path: header line for this journal capture copy.
2. This is the Received: header line constructed during the enqueue from the [process channel](#) to the delivery channel. Note that journal format message copies, being constructed as notification messages, are generated via the process channel.
3. This is the Received: header line constructed by the SMTP server during its enqueue of this capture message to the process channel.
4. The header From: on such a journal message copy is that of the MTA [postmaster](#).
5. The Subject: header line says "Message Journal Copy" for such copies -- unless use of different Subject: text has been configured as discussed in [DSN language and customization](#).
6. The header To: shows the capturing address, as directed in the "capture : journal" action or (new in MS 8.0) the value (sans the ;format-journal tag) of an LDAP attribute named by the [ldap_capture](#) or [ldap_domain_attr_capture](#) or MTA option.

7. New in MS 7.0u4, a X-MS-Journal-Report: header line is generated, as some third-party archive software appears to require such a header line.
8. Note that the journal format consists, at the outermost MIME level, of a multipart/mixed part; this is in contrast to the default, DSN format for captured messages, which, making use of the standard notification message format, consists of a multipart/report part at the outermost MIME level.
9. This text part contains the journal format's minimal set of envelope information for the captured message (itself contained in the subsequent part).
10. The envelope From address for the original message is reported on a "Sender:" line.
11. The Message-id: of the original message is reiterated here.
12. The list of envelope To recipients are reported, one envelope To recipient per line. As of MS 7.0u3, any source routes on the envelope To recipient address or addresses are removed. As of MS 7.0.5, it is the so-called "intermediate address" rather than the "final address" that is reported; this is especially notable for local Message Store recipients. This example corresponds to an original message having only one recipient (or more precisely, an original message having only one recipient at time of capture).
13. This message part contains the original message, essentially as it existed at time of capture, but with the addition of a Return-path: header line, plus a constructed Received: header line contemporaneous with capture processing.
14. The original envelope From address is reported here in a constructed Return-path: header line.
15. A Received: header line is present effectively corresponding to the Received: header line constructed for the original message while capture processing was occurring, though this journal capture copy Received: header line contains no "for ..." clause.
16. From here on, the original message as it existed at time of capture, is duplicated.

Alternatively, again with the same original message, a journal 2007 format message as generated due to a "capture : journal" action when the (new in MS 7.0.5) MTA option [journal_format](#) has bit 0 (value 1) set and when the [addrtypescan](#) channel option has been set on all relevant channels, would have the form shown in [Captured journal 2007 format message copy](#):

Captured journal 2007 format message copy

```
Return-path: <> (1)
Received: from process-daemon.host.domain.com by host.domain.com (Sun Java(tm) (2)
System Messaging Server 7.5-11.01 64bit (built Jul 23 2010)) id
<01NEVSAYXRVK0017PR@host.domain.com> for subpoena1-on-user1@domain.com;
Wed, 28 Jul 2010 17:28:25 -0700 (PDT)
Received: from host.domain.com (Sun Java(tm) System Messaging Server (3)
7.5-11.01 64bit (built Jul 23 2010)) id <01NEVSA2342U0014CT@host.domain.com>;
Wed, 28 Jul 2010 17:28:13 -0700 (PDT)
Date: Wed, 28 Jul 2010 17:28:13 -0700 (PDT)
From: Internet Mail Delivery <postmaster@host.domain.com> (4)
```

```

Subject: Message Journal Copy (5)
To: subpoena1-on-user1@domain.com (6)
Message-id: <0KR2001I49JPZ582@host.domain.com>
MIME-version: 1.0
Original-recipient: rfc822;subpoena1-on-user1@domain.com
X-MS-Journal-Report: (7)
Content-type: multipart/mixed; boundary="Boundary_(ID_xWvHuuTuAMsgGf6g0CDWjg)" (8)

--Boundary_(ID_xWvHuuTuAMsgGf6g0CDWjg)
Content-type: text/plain (9)

Sender: user1@domain.com (10)
Subject: test message
Message-ID: <0IH400G04EU3G800@host.domain.com> (11)
To: <user2@domain.com> (12)

--Boundary_(ID_xWvHuuTuAMsgGf6g0CDWjg)
Content-type: message/rfc822 (13)

Return-path: <user1@domain.com> (14)
Received: from [10.1.110.115] ([10.1.110.115]) (15)
  by host.domain.com (Sun Java(tm) System Messaging Server 7.5-11.01 64bit
  (built Jul 23 2010)) with ESMTP id <0KRH00A3QAO42T10@host.domain.com>;
  Wed, 28 Jul 2010 17:28:11 -0700 (PDT)
Date: Tue, 13 Oct 2009 17:27:03 -0700 (PDT) (16)
Subject: test message
From: user1@domain.com
To: user2@domain.com
Message-id: <0IH400G04EU3G800@host.domain.com>

an original line of text

--Boundary_(ID_xWvHuuTuAMsgGf6g0CDWjg)--

```

In the captured journal format message copy, note the following items of interest:

1. The journal format is a [notification message](#): it has an empty envelope From, as reported here on the Return-path: header line for this journal capture copy.
2. This is the Received: header line constructed during the enqueue from the [process channel](#) to the delivery channel. Note that journal format message copies, being constructed as notification messages, are generated via the process channel.
3. This is the Received: header line constructed by the SMTP server during its enqueue of this capture message to the [process channel](#).
4. The header From: on such a journal message copy is, by default, that of the MTA [postmaster](#). New in MS 7.0.5, if bit 1 (value 2) of the MTA option [journal_format](#) is set, then the From: field value will be set to that of the original message.
5. The Subject: header line normally says "Message Journal Copy" for such copies -- unless use of different Subject: text has been configured as discussed in [DSN language and customization](#), or (new in MS 7.0.5) unless bit 1 (value 2) of the MTA option [journal_format](#) is set (in which case the Subject: field value is set to that of the original message).

6. The header To: normally shows the capturing address, as directed in the "capture : journal" action. New in MS 7.0.5, if bit 1 (value 2) of the MTA option `journal_format` is set, then the To: field value will be set to that of the original message.
7. New in MS 7.0u4, a X-MS-Journal-Report: header line is generated, as some third-party archive software appears to require such a header line. New in MS 7.0.5, if bit 2 (value 4) of the MTA option `journal_format` is set, then an X-MS-Exchange-Organization-Journal-Report: header line is generated rather than an X-MS-Journal-Report: header line.
8. Note that the journal format consists, at the outermost MIME level, of a multipart/mixed part; this is in contrast to the default, DSN format for captured messages, which, making use of the standard notification message format, consists of a multipart/report part at the outermost MIME level.
9. This text part contains the journal format's minimal set of envelope information for the captured message (itself contained in the subsequent part).
10. The envelope From address for the original message is reported on a "Sender:" line. (Note that unlike 2003 format, the envelope From is not enclosed in angle brackets.)
11. The Message-id: of the original message is reiterated here.
12. The list of envelope To recipients are reported, one envelope To recipient per line. As of MS 7.0u3, any source routes on the envelope To recipient address or addresses are removed. As of MS 7.0.5, it is the so-called "intermediate address" rather than the "final address" that is reported; this is especially notable for local Message Store recipients. With bit 0 (value 1) of the MTA option `journal_format` set, as well as optionally also bit 3 (value 8), what exactly will be noted on each envelope To recipient line will depend upon whether the `addrtypescan` channel option has applied, and if so, on whether envelope To recipient address strings *exactly* match recipient header values. This example corresponds to an original message having only one recipient (or more precisely, an original message having only one recipient at time of capture), and where that envelope To recipient string is *exactly* the same as a value on the To: header line. When the `addrtypescan` channel option has been applied, then envelope recipient addresses may be denoted with "To:" (if exactly matching a header To: or Resent-To: value), or denoted with "Cc:" or "Bcc:" similarly, or denoted with simply "Recipient:" if no such exact string match exists; "Recipient:" is also what is used whenever `addrtypescan` was not applied. Recipient addresses may be further elaborated with "Forwarded:" or "Expanded:" labelling, *e.g.*,

`To: active-address, Forwarded: original-address`

in those cases where the MTA can detect that forwarding or list expansion, respectively, has occurred.
13. This message part contains the original message, essentially as it existed at time of capture, but with the addition of a Return-path: header line, plus a constructed Received: header line contemporaneous with capture processing.
14. The original envelope From address is reported here in a constructed Return-path: header line.
15. A Received: header line is present effectively corresponding to the Received: header line constructed for the original message while capture processing was occurring, though this journal capture copy Received: header line contains no "for ..." clause.

16. From here on, the original message as it existed at time of capture, is duplicated.

54.5 Archiving messages

Message archiving can take different forms, and serve different purposes. For record keeping (especially legal compliance) purposes, it may be desired to retain copies of "all" messages that have passed through the Messaging Server, or to retain copies of more specifically selected messages; in this sense, the "archive" is a long term record or history of message traffic, where this historical record of messages is intended for administrative/legal use, rather than (normally) being accessed directly by end e-mail users. For operational efficiency purposes, it may be desired to "archive" messages in the sense of "off-loading" older/larger/less frequently accessed messages to "more economical" storage, that is nevertheless still accessible; in this sense, the "archive" is an implementation-achieved efficiency measure, preferably remaining reasonably conveniently usable by end users.

From the design point of view, these two distinct purposes for archiving mean somewhat different requirements for message access. For compliance archiving, the primary focus is on thoroughness of the scope of message capture, while the requirements for accessing messages are fewer; in particular, whether the message archive is accessible to end users may be optional. While for operational archiving, not all messages may need to be archived---but those messages that are archived must still be reasonably conveniently retrieved by end users.

54.5.1 Choosing which messages to archive

The choice of which messages to archive is a critical one for sites, especially when the archiving is for compliance purposes. This has three components: (1) choosing whose or which types of messages to archive, (2) choosing in what form and at what stage(s) of processing and transitting the MTA the messages should be captured for archiving, and (3) choosing whether [Message Store IMAP APPEND operations \(moving a message to a folder\)](#) should cause archiving. Of these three questions, the first (whose or which types of messages to archive) is usually well specified. The third question (whether to archive due to Message Store operations) also tends to be straightforward to decide. However, the second question may require additional consideration. Between initial message submission and eventual final delivery into a mailbox, while transitting the MTA, messages undergo various transformations, some trivial and some potentially dramatic. Such transformations can include: addition of Received: header lines, addition of other header lines (such as missing-but-required header lines such as Date:, or addition of spam filtering header lines, or addition of mailing list header lines, *etc.*), transformations ("address reversal") of addresses in header lines, alias or list expansion changing the currently active set of envelope recipients, "split up" of a multi-recipient message into different copies for different subsets of recipients, addition of "disclaimer" text, changes in Content-transfer-encoding, document conversion processing, [conversion to a different charset \(CHARSET-CONVERSION\)](#), *etc.*

Three possible approaches for selection of which messages transitting the MTA are eligible for archiving include:

1. Flow-based: Those messages passing through certain channels (such as channels delivering to the Message Store, or channels sending out to the Internet) should be archived.
2. User-based: Those messages sent to or from certain users (perhaps all users; perhaps all users in certain domains; perhaps only some distinguished subset of users) should be archived.
3. Content-based: Those messages containing certain content should be archived.

Such approaches correspond, respectively, to techniques of:

1. For flow-based archiving, it would be typical to trigger archiving via [channel *spamfilter* options](#) (if using an [archiving callout](#) approach) or via channel Sieve filters (using a ["capture" action](#) in a channel Sieve filter located via a [sourcefilter](#) or [destinationfilter](#), as relevant). Choice of the "correct" channels on which to trigger archiving is critical.
2. For user-based archiving, it would be typical to trigger archiving via some user-level (or new in MS 8.0, domain-level) LDAP attribute; see [Capture triggered via LDAP attribute](#). Use of a class-of-service may be helpful in setting such an attribute on all (or large subsets) of users. Note that when such an [ldap_capture](#) or (new in MS 8.0) [ldap_domain_attr_capture](#) named LDAP attribute is used, then capture will occur at whatever channel stage a user alias is expanded (capturing messages to the user), as well as whenever [address reversal](#) occurs (capturing messages from the user). Since address reversal in particular normally occurs during every message enqueue, deployments involving multiple channel "hops" or multiple relay hosts may find multiple "copies" of messages---one "copy" per channel "hop" -- getting captured for archiving. Thus an alternative to such global use of an LDAP attribute is to use instead a [Sieve filter "capture" action](#), perhaps consulting a [Sieve external list](#) (which may consist of consulting a user-level LDAP attribute). This technique of using a channel-specific Sieve filter that consults a Sieve external list allows more precisely timed (limited to specific channels) archiving that is still based on (provisioned via) LDAP attribute settings; see for instance [Example Sieve external lists with properties](#).
3. For content-based archiving, it is critical to detect and label which messages contain the sort of content that needs archiving. If users and user e-mail agents can be relied upon to label such context *ab initio*, when messages are first generated, that is one solution for labelling. Very simple, and easy to detect, content criteria may be codable into a Sieve script---for instance, detecting certain MIME Content-type: labelling. More complex content detection, especially in cases of concerns about uncooperative users attempting to evade archiving requirements, may require special, third-party scanning-and-detection software, a la spam/virus filter software. As usual, the preferred approach for integrating such third party packages is via the MTA's spamfilter plug-in facility; if the third party package does not support such callout use, then the second best choice is to deploy the package "on the side" of the MTA using the usual [aliasdetourhost](#)/alternate conversion channel approach. In any case, once the messages are labelled in whatever way chosen, then the actual trigger for archiving can use [Sieve filter based "capture"](#) triggered by presence of the relevant label.

54.5.2 Message identifier generation

When doing archiving, whether with the AXS:One archiving module or via some other mechanism, it is typically useful or even required to "identify" messages via some identifier. The MTA has the ability to generate an identifying "hash" for messages passing through it, based on configurable selection of message features. It is also (at least typically) necessary to uniquely identify the recipient(s) of the archived message copies---and in some cases, using the recipient's (canonical) e-mail address may not be the preferred choice. The MTA has an option to control the generation of message recipient identifiers. These options will be discussed below.

Messages undergo more-or-less constant change during their traverse from initial submission to eventual final delivery: Received: header lines are added at each "hop"; mailing lists get expanded changing the set of recipients for a message; for a multi-recipient message, the message may need to get "split up" into separate copies for separate types of recipients;

individual addresses undergo routing and esthetic transformations; message contents may be modified as for example addition of header lines indicating spam/virus filtering or content encoding changes or character set conversions, or addition of "disclaimer" text, *etc.* So "identifying" a message is not a straightforward question: one has to ask which portions of a message "matter" as far as the identification is concerned, *vs.* which sorts of "changes" should be ignored. By default, the MTA generates a hash over the following message fields (or a different set may be selected using the `message_hash_fields` MTA option):

- Message-id:
- From:
- To:
- Cc:
- Bcc:
- Resent-message-id:
- Resent-From:
- Resent-To:
- Resent-Cc:
- Resent-Bcc:
- Subject:
- Content-id:
- Content-type:
- Content-Description:

(Note that the Content-type: and Content-description: header are those from the top or outermost MIME level of the message.)

Keep in mind that the choice of when (which channel(s)) to generate a message hash will affect which version(s) of a message are archived. A typical configuration, especially for "operational" mode, might be to set `generatemessagehash` on channels delivering to the Message Store, such as `ims-ms` and `tcp_lmtpcs*` channels, with `deletemessagehash` (the default) on all other channels (especially `tcp_local`). Such a configuration would result in separate message hashes for each message copy separate at the time of enqueue to the final delivery channel. Or in other cases, where it is desired to archive only an "earlier" copy of messages, possibly a copy from before certain later message bifurcations have occurred, then another potentially useful configuration (especially in "compliance" mode) might be to set `generatemessagehash` "earlier", such as on any (known to be used) "intermediate" channels and on channels delivering to other internal hosts such as `tcp_intranet`. Any final delivery channels such as `ims-ms` or `tcp_lmtpcs*` will still need either `keepmessagehash` set (if such messages are guaranteed to have already passed through a channel that did hash generation--as in the case of messages that have already passed through a "front end" MTA) or `generatemessagehash` if it is possible for a message to get to the channel without a previously generated hash. And a channel delivering externally (such as `tcp_local`) should again be set `deletemessagehash`.

The `unique_id_template` MTA option may be used to configure how unique recipient identifiers will be generated for purposes of archiving.

54.5.3 AXS:One archive integration

One approach for message archiving is to make use of an AXS:One archive. The [MTA](#) and the [Message Store](#) support integration with AXS:One for archiving.

54.5.3.1 Architecture of the AXS:One integration

The Messaging Server integration with AXS:One has the following components:

- The MTA and Message Store can compute a [message hash to be used as a unique identifier of the message](#) for AXS:One's purposes.
- Data transfer between Messaging Server (either or both of the MTA and the Message Store) and AXS:One is via configuration-specific file drop directories.

In one such file drop directory, the MTA and Message Store can deposit message copy files: for each message, AXS:One wants an "archive package" consisting of the message main body, any attachment files, and a `.info` file that contains the meta data of the message, the message hash computed by Messaging Server (and stored in the `.info` file's `MessageId` field), and references to the other files in that set (archive package).

Technical note: The Messaging Server writes the `.info` files initially as `.tmp` files, only renaming to `.info` once all the files referenced by that `.info` file are written and the `.info` file itself is complete. That is, until an archive package is complete, the eventual `.info` file will appear instead as a `.tmp` file. On the AXS:One side, it consumes (and deletes) a `.info` file and all the files it references as part of its processing.

In another file drop directory, AXS:One deposits report files to inform Message Server after it has archived messages.

Technical note: AXS:One writes a single line record per message to the report file. When writing, AXS:One writes to a `tmp` file; when the file is completed, it is renamed to `timestamp_mamsg.txt`. (Such files will subsequently be consumed by `imarchive`.)

- The MTA's general "spam filter" plug-in approach can be used to call an AXS:One specific plug-in module. During message enqueues, the MTA can call the AXS:One integration plug-in which generates message copy files (an "archive package") suitable for AXS:One consumption and deposits those message copy files in the integration file drop directory. All the MTA's usual configuration options regarding which messages to copy for archiving are thereby available. As this archiving is occurring while messages are transitting the MTA, it is usually a part of compliance archiving.
- The Message Store's `imexpire` utility has an `archive` action. Expiration rules configured with the `archive` action control which messages are moved to the archive. This is primarily used for operational archiving: for instance, moving older (or larger) messages off to the archive. If the message already has a [message hash identifier for the message](#), then `imexpire` will use it, but otherwise will generate a message hash itself; in either case, `imexpire` will both include the message hash in the archive package it generates for AXS:One, and insert the message hash into the `msghash` database.

-

The Message Store's `imarchive` utility processes the report files generated by AXS:One. `imarchive` should be [scheduled](#) to run periodically to perform this processing task.

Technical note When `imarchive` runs, it renames the `timestamp_mamsg.txt` file generated by AXS:One to `timestamp_mamsg.pid` before processing the records in the file. If such processing is successful, `imarchive` removes the file; otherwise, the file is renamed to `timestamp_mamsg.err`.

54.5.3.2 MTA support for AXS:One archiving

(New JS MS 6.3.) The MTA has support for use with the AXS:One archive facility. AXS:One operates by scanning a directory for files to archive, so the MTA writes copies of the messages to be archived, in a special AXS:One format (in particular including a [hash "identifier"](#) for each copy of a message delivered to the Message Store), to this specified special directory (for AXS:One to then "pick up"). (The AXS:One archive facility will generate a message identifier itself for messages that it finds without one, and this is sufficient for copies of messages that were delivered to external Internet sites. The presence of an identifier generated by the MTA on copies of messages delivered to the Message Store is instead important for correlation of the archived message copies with the message copies actually in the Message Store.)

Note that the MTA's generic facilities to generate and add an "identifier" to messages may be useful with other forms of archiving as well; see [Message identifier generation](#), as well as the [*messagehash](#) channel options, and [Message archival and hashing MTA options](#), for further discussion of the generation and addition to messages of such an identifying "hash" value.

While archiving is most often configured for all users and messages, it is possible to configure archiving only for certain sets of users, using the sorts of "opt-in" approaches (per-user "opt-in" LDAP attribute or channel-level "opt-in") available with the MTA's general spam/virus filter package integration approach (of which the MTA's AXS:One integration is one instance). See the [ldap_optinN](#) MTA options, the [*spamfilterNoptin](#) channel options, and the (new in JS MS 6.3) [ldap_source_optinN](#) MTA options.

When [ldap_source_optinN](#) based triggering of archiving on a per-sending-user basis is desired, note that this is triggered during [address reversal](#), and hence it is critical to be performing address reversal, and in particular properly configured address reversal. See [Intended side effects of LDAP address reversal](#).

54.5.3.3 MTA configuration for AXS:One archiving

From the point of view of MTA configuration, the archiving support is configured rather similarly to configuring use of a third-party, integrated, spam/virus filter package; in particular, use of the archiving support routines is configured via [spamfilterN_config_file](#) and [spamfilterN_library](#) MTA options. For some value of *N*, the `spamfilterN_library` MTA option must be set to point to the `libarch.so` module, while the `spamfilterN_config_file` option points to an archive module option file (in [MTA option file format](#)) controlling a few options for the archiving module. For instance:

```
spamfilter1_library=IMTA_LIB:libarch.so
spamfilter1_config_file=IMTA_TABLE:archive.dat
```

See [Archive_spamfilterN_config_file](#) for further discussion of what may (and must) appear in the archive module option file.

In addition, when it is desired (as it normally is) to be able to correlate the messages in the Message Store with their archived version in the AXS:One system, then the MTA itself should be configured to generate an identifying hash for each message being delivered into the Message Store. (The AXS:One archiving module will generate an identifying message hash for each message that does not already have one itself; this is sufficient for archived messages that do not have a corresponding copy in the Message Store---for instance, archived copies of messages sent to remote recipients.) The material to be hashed, and the algorithm for generating the hash, are controlled via MTA options `message_hash_fields` and `message_hash_algorithm`; channels are individually configurable, via `certain *messagehash channel options`, as to whether they generate, preserve, or delete, such hashes for messages passing through them. See [Message identifier generation](#) for further discussion of the generation of such identifying hashes.

Finally, when archiving messages there is the question of identifying the user for whom messages were originally destined---a user identifier. By default, each user's canonical e-mail address is used. However, the `unique_id_template` MTA option, also discussed in [Message identifier generation](#), may be set to specify use of some alternate form of identifier; when set, the AXS:One archiving facility will generate and use unique user identifiers according to this template, rather than e-mail addresses as by default.

54.5.3.3.1 Example message archiving configuration

Here is an example configuration for using AXS:One archiving.

- In the MTA option file:

```
SPAMFILTER1_LIBRARY=IMTA_LIB:libarch.so
SPAMFILTER1_CONFIG_FILE=IMTA_TABLE:archive.dat
MESSAGE_HASH_ALGORITHM=MD5
!
! Not bothering to hash over Content-type: and Content-description:
!
MESSAGE_HASH_FIELDS=Message-id,Resent-message-id,From,Resent-From,\
  To,Resent-to,Cc,Resent-Cc,Bcc,Resent-Bcc,Subject,Content-id
```

- In the `archive.dat` file:

```
STYLE=1
DIRECTORY=/opt/SUNWmsgsr/archive/
! For compliance mode (legal archiving requirements compliance):
IDSUFFIX=-0000MD500
```

- If the desire is to archive a copy of each message getting delivered to the Message Store on the `ims-ms channel`, plus a copy of each message going out to the Internet on the `tcp_local channel`, then those two channels in the MTA configuration file, `imta.cnf`, would be marked with `destinationspamfilter1`, with the `ims-ms` channel also being marked `generatemessagehash` (so that archived copies could be correlated with the copies in the Message Store, delivered by the `ims-ms` channel), *e.g.*,

```
ims-ms ...rest-of-keywords... generatemessagehash destinationspamfilter1
ims-ms-daemon
```

...other-channel-definitions...

```
tcp_local ...rest-of-keywords... destinationspamfilter1
tcp-daemon
```

Or in Unified Configuration, the MTA and channel options set as:

```
msconfig> set mta.spamfilter1_library IMTA_LIB:libarch.so
msconfig# set mta.spamfilter1_config_file IMTA_TABLE:archive.dat
msconfig# set mta.message_hash_algorithm MD5
msconfig# set mta.message_hash_fields "Message-id Resent-message-id...etc..."
msconfig# set channel:ims-ms.generatemessagehash
msconfig# set channel:ims-ms.destinationspamfilter1
msconfig# set channel:tcp_local.destinationspamfilter1
```

and the `IMTA_TABLE:archive.dat` (*i.e.*, `SERVERROOT/config/archive.dat`) file:

```
STYLE=1
DIRECTORY=/opt/SUNWmsgsr/archive/
! For compliance mode (legal archiving requirements compliance):
IDSUFFIX=-0000MD500
```

Chapter 55 Monitoring the MTA

55.1 MTA transaction logging	55-1
55.1.1 MTA transaction log entry format	55-2
55.1.2 Triggering effects from transaction logging with LOG_ACTION	55-9
55.2 MTA counters	55-20
55.2.1 MTA channel counters	55-20
55.2.2 Purpose and use of MTA counters	55-24
55.2.3 MTA counters implementation	55-24
55.2.4 SNMP subagents	55-24

For monitoring the MTA, usually the best place to start is with the [MTA's optional logging of message traffic](#); from this basic information, sites may gather statistics such as how many messages are passing through the MTA, and answering other questions on message traffic.

The command line utility `imsimta qm` may be used to scan what messages are present in the MTA queue area.

The MTA also has facilities to collect and monitor [channel counters](#) based upon [RFC 1566, the Mail Monitoring MIB](#). Note that counters are intended for providing real-time "snap-shots" of MTA behavior, rather than for gathering the sort of statistics instead available from the log files. For a description of the MTA counters, see the discussion of [MTA counters](#).

The MTA provides utilities to display the counters directly; see the `imsimta counters` and `imsimta qm` utilities, described in [MTA command line utilities](#).

55.1 MTA transaction logging

The MTA's optional logging of message traffic is enabled via the [logging](#) channel option. Enabling `logging` causes the MTA to write an entry to a `mail.log*` file each time a message passes through an MTA channel. Such log entries can be useful if you wish to get statistics on how many messages are passing through the MTA (or through particular channels), or when investigating other questions such as whether and when a message was sent or delivered.

If you are only interested in gathering statistics on the number of messages passing through a few particular MTA channels, then you may wish to enable the `logging` channel option on just those MTA channels of main interest. But more generally, 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 spot such issues.

In addition to the basic information always provided when `logging` is enabled, additional, optional informational fields may also be logged in the `mail.log` files, controlled via various [log_* MTA options](#). Particularly likely to be of interest are the [log_message_id](#), [log_envelope_id](#), [log_filename](#), [log_connection](#), [log_process](#), and [log_username](#) options.

- Enabling [log_message_id](#) allows correlation of which entries relate to which message.
- Enabling [log_envelope_id](#) allows easier correlation of which entries from `mail.log` files from different systems correspond to the same message at the SMTP level.

- Enabling `log_filename` makes it easier to immediately spot how many times delivery of a particular message file has been retried, and can be useful in understanding when the MTA does or does not split a message to multiple recipients into separate message file copies on disk.
- Enabling `log_connection` causes the MTA to log TCP/IP connections, as well as message traffic, to the `mail.log` files by default; alternatively, the `separate_connection_log` option may be used to specify that connection log entries instead be written to `connection.log` files.
- When using `log_connection` to cause generation of TCP/IP connection entries, additionally enabling `log_process` allows correlation of which connection entries correspond to which message entries.
- `log_username` is mostly of interest in the case of users who authenticate their message submission using the SMTP AUTH command; in such cases, `log_username` causes logging of the user identity (prefixed with an asterisk character if the identity was established by authentication). As of the 8.0 release, the `log_username` option can also be used to log the primary mail address associated with the authenticated identity and in the case of "U" connection log entries, the authentication mechanism used.

On UNIX, `mail.log` and `connection.log` entries may optionally be duplicated to syslog via the `log_messages_syslog` and `log_connections_syslog` options.

55.1.1 MTA transaction log entry format

The format of message transaction log entries and connection transaction log entries is subject to change. By default, message transaction log entries and connection transaction log entries all appear in the same message log files (`mail.log*` files); however, if the MTA option `separate_connection_log=1` is set, then the connection transaction log entries will instead appear in the connection transaction log files (`connection.log*` files).

Currently, by default, each message transaction log entry contains eight or nine fields, *e.g.*,

```
19-Jan-1998 19:16:57.64 tcp_intranet tcp_local E 1 adam@domain.com rfc822;mark@innosoft.com mark@innosoft.com
(1)           (2)           (3)           (4) (5)           (6)           (7)           (8)           (9)
```

These fields are:

1. The date and time when the entry was made.
2. The channel name for the source channel.
3. The channel name for the destination channel.
4. The type of entry; see [Message logging entry action type codes](#).
5. The size of the message.¹ This is expressed in kilobytes by default, although this default can be changed by using the `block_size` MTA option. If message size is not an exact `block_size` multiple, then the size is rounded up to the next block for logging purposes. (Note that in "Q" records, the size is not necessarily the size of the message as a whole, but rather indicates the amount of message processed before the delivery attempt failed: in particular, the size field in a "Q" record may be 0 such as in cases where the MTA's SMTP client encounters a connection failure, or the size field may correspond to the full size of the

message such as in cases where the MTA's SMTP client encounters message rejection after the final ".", or in cases such as a network disconnect part way through message transfer the size field will indicate roughly at what point in message transfer the disconnect occurred.)

6. The envelope From address. Note that for messages with an empty envelope From address, such as notification messages, this field will be blank.
7. The original form of the envelope To address. (Note that this is the ORCPT value, and hence follows ORCPT syntax; see [RFC 3461](#). Also note that the semantics of ORCPT are neither "originally submitted address", nor "address originally given to this MTA", and hence ORCPT only sometimes corresponds to one or the other or both. For the "address originally given to this MTA", see instead the [log_intermediate](#) MTA option.)
8. The active (current) form of the envelope To address.
9. The delivery status (SMTP channels only).

Table 55.1 Message logging entry action type codes

Type	Modifiers	Description
General		
B	E, P, A, S, U, B, C	(New in JES MS 6.2) Unrecognized SMTP command.
D	E, L, A, S, U, B, C	Successful dequeue
E	E, P, A, S, U, B, C	Enqueue. An "E" or null modifier will be present to indicate whether the enqueue was with EHLO or HELO, respectively.
J	E, P, A, S, U, B, C	Rejection of attempted enqueue (rejection by slave channel program); as of MS 7.0, routinely generated for LMTP server rejections (whereas in previous versions only LMTP rejections at the envelope address stages would be recorded, while LMTP rejections after the DATA would not be recorded), though LMTP server rejections do not record the full range of modifiers (see below).
K	E, L, S, U, B, C	Recipient address rejected on attempted dequeue (rejection by master channel program) when the recipient has the NOTIFY=NEVER DSN flag set (so no bounce message will be generated regarding this rejection), or deletion of a timed-out NOTIFY=NEVER message by the return job; compare with "R" records which are the same sort of rejection/time-out occurring, but where a new notification message is also generated regarding this failed message
P	F, X, J, Y, D	(New in MS 8.0) Request to generate a Delivery Status Notification . (Note that in some cases no actual DSN will end up being sent.) "P" records are only generated in cases where the error causing the DSN isn't recorded in any other log entry. The various cases where this happens are further detailed by the presence of a modifier character on the action. Currently the defined modifiers are: <ul style="list-style-type: none"> • F - address errors detected during alias or mailing list expansion operations (this includes bad RCPT TO addresses that are allowed because the acceptalladdresses channel option is in effect) • X - capture operations

		<ul style="list-style-type: none"> • J - journal operations • Y - Sieve syntax or evaluation errors • D - success delivery receipts
Q	E, L, S, U, B, C	Temporary failure to dequeue
R	E, L, S, U, B, C	Recipient address rejected on attempted dequeue (rejection by master channel program), or generation of a failure/bounce message; compare with "K" records, which are the same sort of rejection/time-out occurring, but where no notification message is generated and the original message is just deleted
V	E, L, P, A, S, C	(New in JES MS 6.2p2) An incoming SMTP transaction ended prematurely; frequently corresponds to an address verification operation by a remote SMTP client. The "C" modifier was not recorded on "V" records until circa MS 7.0u4
W		Warning message (generated by the return_job) regarding a not-yet-delivered message
Z	E, L, P, A, S, U, B, C	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 reenqueued
LMTP server		
J	S, C	Rejection by LMTP server of address or message; as of MS 7.0, routinely generated for LMTP server rejections (whereas in previous versions only LMTP rejections at the envelope address stages would be recorded, while LMTP rejections after the DATA would not be recorded). Prior to MS 7.0u4, note that unlike SMTP and SMTP SUBMIT server "J" records, LMTP server "J" records did not include any modifier letters; new in MS 7.0u4, "S" (TLS used) and/or "C" (CHUNKING used) modifiers may be present.
S	S, C	(New in MS 7.0) LMTP deposit into the store; in prior versions indicated with the (somewhat misleading) D action code instead; the addition of "S" (TLS used) and/or "C" (CHUNKING used) modifiers is new in MS 7.0u4.
SMTP/LMTP modifiers		
	E	(New in MS 6.3) ESMTP (EHLO) used
	L	(New in MS 6.3) LMTP (LHLO) used
		Absence of both "E" and "L" implies that HELO was used
	P	(New in MS 7.0) POP-before-SMTP used (via the MMP)
	Q	(New in 7.0) Pipelining used
	A	Authentication (SMTP AUTH) successfully used
	S	TLS (STARTTLS) successfully used
	U	(New in MS 7.0) BURL used.
	B	(New in MS 8.0) BINARYMIME used; see the binaryserver channel option.
	C	(New in MS 6.3) CHUNKING used without BINARYMIME; see the chunking* channel options.

In addition to the default message transaction fields (shown above), the MTA may optionally be configured to log additional information to the message transaction log file; see the `log_*` MTA options described in [Transaction logging MTA options](#). With `log_connection`, `log_filename`, `log_envelope_id`, `log_message_id`, `log_node`, `log_notary`, `log_sensitivity`, `log_priority`, `log_process`, and `log_username` all enabled, the format becomes as follows. (Note that the sample transaction log entry line has been wrapped for typographic reasons; the actual message transaction log entry would appear on one physical line.)

```
19-Jan-1998 13:13:27.10 hosta  2e2d.5.1 tcp_local  tcp_intranet E 1 service@innosoft.com
      (1)          (10)    (11)    (2)      (3)      (4) (5) (6)

rfc822;adam@domain.com adam 276
(7)          (8) (12)

/opt/sun/comms/messaging64/data/queue/tcp_intranet/ZZi0D4d9f5mwC.00
(13)

<01IWFVYLGTS499EC9W@innosoft.com> <01IWFVYLGTS499EC9Y@innosoft.com>
(14)          (15)

mailsrv innosoft.com (innosoft.com [192.160.253.66]) 0 3
(16) (17)          (18) (19) (9)
```

Here the additional fields, beyond those already discussed [above](#), are:

- (10) The name of the node on which the channel process is running.
- (11) The process id (expressed in hexadecimal), followed by a period (dot) character, if it is a multithreaded channel entry a process id and another period (dot), and finally a count.
- (12) The NOTARY (delivery receipt request) flags for the message, expressed as an integer.
- (13) The file name in the MTA queue area.
- (14) The envelope id.
- (`log_tracking` new in MS 8.0 and not shown in the above example) Tracking ID.
- (`log_times` new in MS 8.0 and not shown in the above example) Deferred delivery time and expiry time.
- (15) The message id.
- (16) The username of the executing process. Note that in the case of Dispatcher services such as the SMTP server, this will be the username of the user who most recently did a startup of the Dispatcher.
- (`log_auth` new in MS 7.0.5 and not shown in the above example) The SMTP MAIL FROM's AUTH parameter value.
- (17) The exact connection information shown varies according to whether a message is incoming (E record) or outgoing (*e.g.*, D record), whether or not the channel is an SMTP (or LMTP) channel, and for SMTP/LMTP channels for incoming messages, the specific bits set for the `log_connection` MTA option. For incoming messages, 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 options. This sample assumes use of one of these options, for instance use of the default `identnone` channel option, that selects display of both the name found from the DNS and IP address. This example also assumes that `log_connection=1` is set, but that higher bits of `log_connection` are not set. If bit 5 (value 32) of `log_connection` were set, then the incoming connection information for a message incoming over TCP/IP would also include the entire transport information string, `TCP|MTA-IP|MTA-port|remote-IP|remote-port`. If bit 6 (value 64) of `log_connection` were set, then the incoming connection information would also include the application information string, just `SMTP` for the case of incoming SMTP messages. For outgoing messages, *e.g.*, `D` records, the connection information (due to `log_connection`'s bit 0/value 1 being set) is present only for SMTP/LMTP channels, and in such cases consists of the remote host name and the remote name as found in the DNS, the transport information string (see above), and the remote SMTP banner line. And this information is included at the start of the SMTP diagnostic field.

- (18) The sensitivity for the message.
- (`log_mtpriority` new in MS 8.0 and not shown in the above example) The SMTP `MT-PRIORITY` associated with the transaction.
- (19) This effective processing priority for the message; 3 corresponds to "normal" priority. Note that the effective processing priority may not be the same as the message's `Priority`: header value (if any); for instance, the `*blocklimit` channel options can cause lowering of effective message processing priority.
- (`log_intermediate` not shown in the above example) The intermediate form of the recipient address.
- (`log_intermediate` not shown in the above example) The original (`RCPT TO`) form of the recipient address.
- (`log_uid` new in MS 8.0 and not shown in the above example) LDAP `uid` attribute for local users.
- (`log_mailbox_uid` new in MS 7.0.5 and not shown in the above example) For messages delivered to the MS Message Store, the `UID` and `UIDVALIDITY`.
- (`log_futurerelease` new in MS 8.0 and not shown in the above example) SMTP `FUTURERELEASE` value.
- (`log_filter` not shown in the above example) The `Sieve filter` actions applying to the message, including effects from verdicts from spam/virus package "plug-ins".
- (`log_reason` new in JES MS 6.3 and not shown in the above example) The `reason` field (due to setting `log_reason=1`). It would appear in a message transaction log entry corresponding to a message rejection (for instance, an "R" or "K" entry), appearing just before the SMTP delivery status (SMTP diagnostic) field.
- (`log_diagnostics` not shown in the above example) The SMTP delivery status/SMTP diagnostic field (due to having the default of `log_diagnostics=1` set)
- (`log_queue_time` new in MS 6.3 and not shown in the above example) The "time in queue" field (due to setting `log_queue_time=1`).
- (`log_conversion_tag` new in MS 7.0.5 and not shown in the above example) Any `conversion tags` on the message.
- (`log_imap_flags` new in MS 7.0.5 and not shown in the above example) Any IMAP flags that have been set on the message by the MTA.

- ([log_delivery_flags](#) new in MS 7.0.5 and not shown in the above example) Delivery flags.
- ([log_callout_delays](#) new in MS 8.0 and not shown in the above example) Callout delay timer values.
- ([log_transactionlog](#) new in MS 8.0 and not shown in the above example) String(s) logged due to the Sieve "transactionlog" action.

The maximum line length for message transaction records is 4096 characters.

Currently, each connection transaction log entry contains at least six fields, with the presence of up to five additional optional fields controlled by the MTA options [log_node](#), [log_process](#), [log_message_id](#), [log_username](#), and [log_queue_time](#), *e.g.*, (note that for display purposes here, the output lines have been wrapped at the transport information field at (7)),

```
04-Sep-2002 01:00:04.23 host.domain.com 1f625.d.0 tcp_local      +          O
                    TCP|129.153.12.42|25|123.4.5.67|65228 SMTP
04-Sep-2002 01:00:05.21 host.domain.com 1f625.d.3 tcp_local      +          C
                    TCP|129.153.12.42|25|123.4.5.67|65228 SMTP/TLS-192-DES-CBC3-SHA
04-Sep-2002 01:00:06.23 host.domain.com 1f627.3.0 tcp_local      -          O
                    TCP|129.153.12.42|4303|123.45.6.7|25 SMTP/domain.com/mail.domain.com
04-Sep-2002 01:00:06.49 host.domain.com 1f627.3.3 tcp_local      -          C
                    TCP|129.153.12.42|4303|123.45.6.7|25 SMTP/domain.com/mail.domain.com/TLS-192-DES-CBC3-SHA

(1)                (2)                (3)        (4)        (5)        (6)
                    (7)                (8)
```

- (1) The date and time when the entry was made.
- (2) [Optional---only present when [log_node=1](#) is set.] (New in JES 5/MS 6.3) The host name of the MTA system.
- (3) [Optional---only present when [log_process=1](#) is set.] The process id (expressed in hexadecimal), followed by a period (dot) character and then a thread id, followed by a period (dot) character and a count.
- (4) The channel name for the source channel. In the case of "-" entries (outgoing messages), this is the name of the channel acting as SMTP client or LMTP client. Note, however, that in the case of "+" entries (incoming messages), the name shown is that of the default channel for the Dispatcher service listening on the port and interface address for the incoming connection, so is typically merely one of `tcp_local` (for the SMTP server on port 25) or `tcp_submit` (for the SMTP SUBMIT server on port 587) or `tcp_lmtpss` (for the LMTP server); in particular, channel "switching" due to [*switchchannel](#) channel option based effects (such as switching to a `tcp_intranet` channel due to `switchchannel` or switching to a `tcp_auth` channel due to `saslswitchchannel`) is not reflected in such entries. (In the case of "I" records, that is, ETRN records, this "source" channel field instead is used to display the name of the channel which the ETRN command would cause to run.)
- (5) A plus, +, or minus, -, indicating whether this is an inbound or outbound connection, respectively. That is, a + indicates a connection inbound to an SMTP, SMTP SUBMIT, or LMTP server; a - indicates a connection outbound by a channel acting as an SMTP (or LMTP) client.
- (6) A code indicating the type of entry; see [Connection logging entry action type codes](#).
- (7) The transport information. This takes the form:

TCP | *local-IP* | *local-port* | *remote-IP* | *remote-port*

- (8) The application information. For inbound connections (to the SMTP server), the "O" (that is, open) records will just show "SMTP" in this field; the "C" (that is, close) records will just show "SMTP" unless TLS was used, in which case this field will show "SMTP/TLS-info". The TLS-info string consists of "TLS-bitstrength-cipherinfo". (Note that the cipherinfo field may not be present, and if present may be unreliable in the JES MTA, especially as of JES MS 6.0 and later, as the cipher information is not reliably reported back by the underlying NSS library in use.) For outbound connections, the field has some additional information, showing the initial host name (prior to DNS lookup) to which to connect, and the host name found from doing a DNS lookup, that is, the host name to which the connection was really made/attempted. In the case of outbound connections where TLS was used, the TLS information will also be shown in the C (that is, close) record. So for outbound connections, the field takes the form

SMTP/*initial-host*/*DNS-host*

or when TLS was used, the C records will take the form

SMTP/*initial-host*/*DNS-host*/*TLS-info*

For instance, *initial-host* might be a name used for e-mail addresses which merely has MX records, and then *DNS-host* will be the actual host name to which the connection was made (the name pointed to by an MX record).

- (9) [Optional---only present if `log_message_id=1`.] In "T" records, the host name presented on the ETRN command line. In "U" records, the **MTA AUTH error**, if there was one.
- (10) [Optional---only present if `log_username=1`.] In "U" records, the authenticated user.
- (11) (New in MS 6.2) "C" records may include additional information about the reason for the close, if the close was due to an error. For instance, "Error reading SMTP packet" (in cases where the connection was dropped, for instance due to a network problem or the remote system aborting the connection), or "Timeout after *x* minutes trying to read SMTP packet" (in cases where the MTA times out the connection due to remote system inactivity).
- (12) [Optional---only present if `log_queue_time=1`.] (New in MS 6.3) "O" records may include the "time to open the connection" or "Y" records may include the "time attempting to open a connection for an attempt that failed" as a final field.

Table 55.2 Connection logging entry action type codes

Type	Modifiers	Description
Actions		
C	F	Connection closed
O		Connection opened
T		PORT_ACCESS mapping table rejection; logged if both bit 1 (value 2) of the <code>log_connection</code> MTA option is set (or overridden by the <code>LOG_CONNECTION</code> TCP/IP-channel-specific option), and the <code>\$T</code> flag is used in a <code>PORT_ACCESS</code> rejection entry

U		(New in JES MS 6.2) Authentication attempt (SMTP AUTH use), whether successful or failed; logged if bit 5 (value 32) of the <code>log_connection</code> MTA option is set
X	F	Connection rejected, or closed, due to an SMTP level error response
Y		Connection try failed before being established
I		ETRN command received; logged if bit 2 (value 4) of the <code>log_connection</code> MTA option is set (or overridden by the <code>LOG_CONNECTION</code> TCP/IP-channel-specific option)
Modifiers		
	F	(New in MS 7.1) A *.data-failed file was created; may be reported on "C" or "X" records for SMTP and SMTP SUBMIT connections (but not for LMTP connections, since *.data-failed files are never generated by the LMTP server)

The maximum line length for connection transaction records is 4096 characters.

¹The mechanism for computing size values in enqueue entries in the MTA transaction logs was revised for MS 7.0 update 2. Previously message sizes were computed based on counting octets in the input, which did not take various things, including charset-conversion, into account. It was done this way in order to facilitate certain calculations needed for performing [message fragmentation](#). Now that message fragmentation has become a rarity, this approach is no longer appropriate, and the code has been changed to work directly with the output message.

55.1.2 Triggering effects from transaction logging with LOG_ACTION

(LOG_ACTION itself was added in Messaging Server 7.0-3.01. But use of LOG_ACTION with [MeterMaid](#)---other than simple "throttle" calls -- typically requires MeterMaid features new in Messaging Server 7.2-0.01. In particular, MeterMaid "remove" and "test" routines are new in Messaging Server 7.2-0.01.)

The LOG_ACTION [mapping table](#) provides a way for any transactions recorded by the MTA to also, as a side-effect, trigger other effects. A great deal of information, of different types, can be reported in the [MTA's message transaction and connection transaction log files](#). Sites may be interested in noticing certain sorts of log entries as evidence of certain sorts of occurrences, or counting (or at least monitoring trends for) certain sorts of occurrences, or making access decisions based on certain sorts of occurrences. The LOG_ACTION mapping table provides a way to turn MTA message transaction and connection transaction log file entries into syslog notices, or into MeterMaid counter updates; or if a site wishes to provide their own routine for the mapping table to call, to take whatever, site-defined "action" the site chooses, based upon relevant transaction log entries. For instance, a site might want to notice (via a syslog notice) failed SMTP AUTH attempts as a warning of possible account break-in attempt; or a site might want to count (via MeterMaid) the number of failed (bad) recipients for users' outgoing messages, and react to "high" numbers as a possible sign of a user sending spam with a poor-quality recipient list.

55.1.2.1 LOG_ACTION operation

The LOG_ACTION [mapping table](#) is probed each time a [message transaction or connection transaction log entry](#) is written. The LOG_ACTION mapping table's only direct effect on the

normal transaction log entries is its ability to disable output (recording) of specified entries. But its more interesting uses tend to be for its "side effects", which can be considered rather similar to the "side effects" available for the [address based * _ACCESS mapping tables](#) and [FROM_ACCESS mapping table](#): In particular, LOG_ACTION has the potential to generate syslog notices, or make call-outs such as to [MeterMaid](#).

55.1.2.2 Probe format

The format of the LOG_ACTION probe for a message transaction log entry consists at a minimum of the following, plus additional, optional fields:

```
source-channel|destination-channel|action|size|envelope-from|orig-envelope-to|current-envelope-to
```

Here *action* is the usually logged action code. Additional log entry fields may be included in the probe, depending upon the setting of the corresponding [log_* MTA options](#); usually bit 1 (value 2) for a *log_** MTA option controls whether the field controlled by that option is included in the LOG_ACTION probe. For [log_connection](#), where bit 1 already has another meaning, bit 8 (value 256) enables inclusion of the claimed source system (as claimed in the client's the HELO/EHLO command for SMTP channels, or the enqueueing channel's official host name for other types of channels), and the bit that enables inclusion of *application-info* and *transport-info* in the LOG_ACTION probe is bit 9 (value 512); for [log_intermediate](#), bits 2 and 3 (values 4 and 8, respectively) control the inclusion of fields in the probe. These optional fields consist of:

```
|notary-bits|filename|envelope-id|message-id|username|source-system|sensitivity  
|mt-priority|priority|intermediate-dest|initial-dest|ldap-uid|imap-uid:uidvalidity  
|future-release|filter|reason|diagnostics|application-info|transport-info  
|time-in-queue|conversion-tags|imap-flags|delivery-flags
```

The *source-system*, and *application-info* and *transport-info* fields result from two bits of [log_connection](#). The *intermediate-dest* and *initial-dest* fields result from two bits of [log_intermediate](#). The rest of the fields result from, respectively, [log_notary](#), [log_filename](#), [log_envelope_id](#), [log_message_id](#), [log_username](#), [log_sensitivity](#), [log_mtpriority](#), [log_priority](#), [log_uid](#), [log_mailbox_uid](#), [log_futurerelease](#), [log_filter](#), [log_reason](#), [log_diagnostics](#), [log_queue_time](#), [log_conversion_tag](#), [log_imap_flags](#), and [log_delivery_flags](#).

The format of the probe for a connection transaction log entry consists at a minimum of the following, plus additional, optional fields:

```
source-channel|direction|action
```

The *direction* is either + for inbound connections, or - for outbound connections. *action* is the usually logged connection action code, with the possible addition of an "F" suffix (on "C" or "X" action entries), added if the entry corresponds to a case where the MTA encountered an error with its attempt to create a *.data-failed file. The optional fields consist of any subset of:

```
|SASL-error-or-ETRN-host-name|username|diagnostics|transport-info|application-info|queue-time
```

Optional fields are enabled by the relevant bit or bits of [log_message_id](#) (bit 1, value 2 enables logging of the SASL error in "U" SMTP AUTH records and the host name from client ETRN commands in "I" ETRN records), [log_username](#) (especially relevant for U SMTP AUTH records), [log_diagnostics](#), [log_connection](#) (bit 9, value 512 enables both *transport-info* and *application-info*), and [log_queue_time](#). Note that the

same bit (or bits) enable probe inclusion both for message transaction probes and connection transaction probes.

If the probe string matches the pattern (*i.e.*, the left hand side of an entry in the mapping table), then the resulting output template (right hand side) is checked. The output templates in the LOG_ACTION mapping table can use the special flags defined in the table below, as well, of course, as any general mapping table substitutions or metacharacters such as calling out to a routine.

Table 55.3 LOG_ACTION mapping flags

Flag	Description
\$F	Disable writing this entry to the transaction log file
\$N	Disable writing this entry to the transaction log file
\$<string	Send <i>string</i> as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if probe matches; see also the sndopr_priority MTA option
\$>string	Send <i>string</i> as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if access is rejected; see also the sndopr_priority MTA option

55.1.2.3 Examples of LOG_ACTION use

This section shows examples of some possible uses of the LOG_ACTION mapping table. The syntax and general operation of the LOG_ACTION mapping table is discussed above. The first example below is a very simple use to disable recording of certain entries. The additional LOG_ACTION examples below are more sophisticated, making use of [MeterMaid](#) callouts.

55.1.2.3.1 Disabling logging of connections from a periodic monitoring source

One use of LOG_ACTION is to disable logging of some particular type of entry, while still retaining logging in general: for instance, connection attempts from some special source, when such connections are performed merely for "monitoring" or "heartbeat" reasons, may not deserve to be recorded.

For instance, if the monitor source IP is `monitor-ip`, then set bit 9 (value 512) of the [log_connection](#) MTA option and then use a LOG_ACTION mapping table along the lines of that shown below to disable the logging of the connection "O"pen and connection "C"lose records generated by the monitoring probe connections.

LOG_ACTION

```
* | * | O | * | monitor-ip | *      $N
* | * | C | * | monitor-ip | *      $N
```

55.1.2.3.2 Syslog notices after SMTP AUTH attempts with bad password

One use of LOG_ACTION might be to generate a syslog notice if more than some site-chosen number of bad password SMTP AUTH attempts are made against any user account. This can be achieved by calling out to MeterMaid from LOG_ACTION, and then generating the syslog notice when MeterMaid's threshold is reached.

First, in the MTA option file make sure that appropriate data will be included in the LOG_ACTION mapping table probes by setting `log_* MTA options` as below, and set `sndopr_priority` to values for syslog facility and severity that will coordinate with your `syslog.conf` configuration for syslog notice handling:

```
# msconfig
msconfig> show log_connection
role.mta.log_connection = 6
instance.channel:tcp_monitor.log_connection = 0
msconfig> set mta.log_connection 134
msconfig# write -remark="Set bit 7/value 128 of log_connection to get U connection records"
msconfig> set log_message_id 3
msconfig# set log_username 3
msconfig# set log_diagnostics 3
msconfig# write -remark="Enable more fields in LOG_ACTION probes"
msconfig> set sndopr_priority 20
msconfig# write -remark="MTA syslog notices to get LOG_MAIL+LOG_WARNING syslog.conf handling"
msconfig> quit
#
```

In legacy configuration mode, this would correspond to setting MTA options in the `option.dat` file along the lines of:

```
! Sites likely want additional bits of LOG_CONNECTION set; this example
! requires that bit 7/value 128 be set.
!
LOG_CONNECTION=128
LOG_MESSAGE_ID=3
LOG_USERNAME=3
LOG_DIAGNOSTICS=3
!
! Set SNDOPR_PRIORITY to a syslog facility+severity value that will
! coordinate with your syslog.conf configuration, e.g.
! SNDOPR_PRIORITY=20 to choose LOG_MAIL+LOG_WARNING syslog.conf handling.
!
SNDOPR_PRIORITY=20
```

(The above settings represent likely site practice, rather than what is strictly required, as they show `log_*` option values set so that regular MTA connection transaction log entries will be generated as well as fields in LOG_ACTION probes; but in principle, the LOG_ACTION probes could be set without having to enable the connection transaction logging.)

To have a [MeterMaid](#) table named `bad_password_attempts`, configure MeterMaid with a new table via:

```
# msconfig
msconfig> ! First check if MeterMaid has had basic configuration:
msconfig> show metermaid.*
role.metermaid_client.server_host = host-name
role.metermaid.enable = 1
role.metermaid.secret (suppressed)
msconfig> ! Yes, MeterMaid basics already configured;
msconfig> ! so now add a new bad_password_attempts table
msconfig> set metermaid.local_table:bad_password_attempts.data_type string
msconfig# set metermaid.local_table:bad_password_attempts.max_entries 1000
msconfig# set metermaid.local_table:bad_password_attempts.table_options "nocase penalize"
msconfig# set metermaid.local_table:bad_password_attempts.table_type throttle
```

```
msconfig# set metermaid.local_table:bad_password_attempts.quota 5
msconfig# set metermaid.local_table:bad_password_attempts.quota_time 3600
msconfig# write -remark="Added bad_password_attempts MeterMaid table"
msconfig> quit
#
```

In legacy configuration mode, (assuming basic MeterMaid configuration had already been performed via additional configutil parameters not shown below), this would correspond to a MeterMaid table configured via configutil values, including these (though note that many of the values shown being set are actually defaults):

```
metermaid.table.bad_password_attempts.data_type: string
metermaid.table.bad_password_attempts.max_entries: 1000
metermaid.table.bad_password_attempts.options: nocase,penalize
metermaid.table.bad_password_attempts.type: throttle
metermaid.table.bad_password_attempts.quota: 5
metermaid.table.bad_password_attempts.quota_time: 3600
```

Then a LOG_ACTION mapping table to make use of that MeterMaid table could be as follows:

LOG_ACTION

```
! With log_connection=128 set, we get "U" action records.
! With log_message_id=3 set, the SASL-error is recorded in the message-id field
! With log_username=3, get a username field
! With log_diagnostics=3, get a diagnostics field
!
! So probe format is:
!
! source-chan|direction|action|SASL-error|username|diagnostics
!
tcp_*|+|U|Bad$ password$_*|$_*|* \
$[IMTA_LIB:check_metermaid.so,throttle,bad_password_attempts,$1]$<LOGACTION,$ \
Too$ many$ bad$ password$ attempts$ for$ user$ $1
```

There is a subtlety in the above, which is that `log_diagnostics` is shown being set *purely* to ensure that there is at least one more vertical bar and (possibly empty) field appearing after the username field. Furthermore, asterisk wildcard for matching the username field has the "minimal matching" `$_*` modifier set on it.) This is not strictly necessary for this *exact* example, but makes this example easier to extend with additional fields should sites wish to do so.

There is also a prior `$_*` match in the reason field portion of the pattern, just after "Bad\$ password"; that is necessary as of 8.0 when additional detail text was added to the reason field.

55.1.2.3.3 Syslog notices after SMTP AUTH attempts with bad username

Another example would be to generate a syslog notice if the same source IP makes multiple attempts to authenticate with a bad username (hence suggestive of a possible "dictionary attack" against your user name space). With MTA options settings of:

```
# msconfig
msconfig> show log_connection
role.mta.log_connection = 2
msconfig> set mta.log_connection 642
```

Triggering effects from transaction logging with LOG_ACTION

```
msconfig# write -remark="Set bit 1/value 2 plus bit 7/value 128 plus bit
9/value 512 of mta.log_connection to get O and C plus U connection records,
plus application and transport fields included in LOG_ACTION probes"
msconfig> set log_message_id 3
msconfig# set log_username 3
msconfig# set log_diagnostics 3
msconfig# write -remark="Enable extra fields in LOG_ACTION probes"
msconfig# set sndopr_priority 20
msconfig# write -remark="MTA syslog notices to get LOG_MAIL+LOG_WARNING syslog.conf handling"
msconfig> quit
#
```

Or in legacy configuration mode:

```
! Sites likely want additional bits of LOG_CONNECTION set; this example
! requires that bit 1/value 2 plus bit 7/value 128 plus bit 9/value 512 be set.
LOG_CONNECTION=642
LOG_MESSAGE_ID=3
LOG_USERNAME=3
LOG_DIAGNOSTICS=3
!
! Set SNDOPR_PRIORITY to a syslog facility+severity value that will
! coordinate with your syslog.conf configuration, e.g.
! SNDOPR_PRIORITY=20 to choose LOG_MAIL+LOG_WARNING syslog.conf handling.
!
SNDOPR_PRIORITY=20
```

To have a MeterMaid table named `bad_user_attempts`, configure MeterMaid with a new table via:

```
# msconfig
msconfig> ! First check if MeterMaid has had basic configuration:
msconfig> show metermaid.*
role.metermaid_client.server_host = host-name
role.metermaid.enable = 1
role.metermaid.secret (suppressed)
msconfig> ! Yes, MeterMaid basics already configured;
msconfig> ! so now add a new bad_user_attempts table
msconfig> set metermaid.local_table:bad_user_attempts.data_type ipv4
msconfig# set metermaid.local_table:bad_user_attempts.max_entries 1000
msconfig# set metermaid.local_table:bad_user_attempts.table_options "penalize"
msconfig# set metermaid.local_table:bad_user_attempts.table_type throttle
msconfig# set metermaid.local_table:bad_user_attempts.quota 5
msconfig# set metermaid.local_table:bad_user_attempts.quota_time 3600
msconfig# write -remark="Added bad_user_attempts MeterMaid table"
msconfig> quit
#
```

In legacy configuration mode, (assuming basic MeterMaid configuration had already been performed via additional `configutil` parameters not shown below), this would correspond to a MeterMaid table configured via `configutil` values including:

```
metermaid.table.bad_user_attempts.data_type: ipv4
metermaid.table.bad_user_attempts.max_entries: 1000
```

```
metermaid.table.bad_user_attempts.options: penalize
metermaid.table.bad_user_attempts.type: throttle
metermaid.table.bad_user_attempts.quota: 5
metermaid.table.bad_user_attempts.quota_time: 3600
```

then a LOG_ACTION mapping table could be:

LOG_ACTION

```
! With log_connection=642 set, we get "U" action records and transport-info
! fields.
! With log_message_id=3 set, the SASL-error is recorded in the message-id field
! With log_username=3, get a username field
! With log_diagnostics=3, get a diagnostics field
!
! So probe format is:
!
! source-chan|direction|action|SASL-error|username|diagnostics|transport-info
! where transport-info is: TCP|host-IP|host-port|source-IP|source-port
!
tcp_*|+|U|No$ such$ user|*|*|TCP|$_*|$_*|$_*|* \
$[IMTA_LIB:check_metermaid.so,throttle,bad_user_attempts,$5]$<LOGACTION,$ \
Too$ many$ wrong$ username$ attempts$ from$ $5$ (username$ attempted:$ $1)
```

55.1.2.3.4 Syslog notices after failing SMTP AUTH attempts, resetting after success

Both of the above examples could be improved by using the (new in 7.2-0.01) remove routine of MeterMaid to achieve a "reset" effect after desired "good" occurrences. For instance, with settings as above (including log_diagnostics=3 and log_connection=642):

LOG_ACTION

```
! With log_connection=642 set, we get "U" action records and transport-info
! fields.
! With log_message_id=3 set, the SASL-error is recorded in the message-id field
! With log_username=3, get a username field
! With log_diagnostics=3, get a diagnostics field
!
! So probe format is:
!
! source-chan|direction|action|SASL-error|username|diagnostics|transport-info
! where transport-info is: TCP|host-IP|host-port|source-IP|source-port
!
tcp_*|+|U|Bad$ password|$_*|* \
$[IMTA_LIB:check_metermaid.so,throttle,bad_password_attempts,$1]$<LOGACTION,$ \
Too$ many$ bad$ password$ attempts$ for$ user$ $1
tcp_*|+|U|No$ such$ user|*|*|TCP|$_*|$_*|$_*|* \
$[IMTA_LIB:check_metermaid.so,throttle,bad_user_attempts,$5]$<LOGACTION,$ \
Too$ many$ wrong$ username$ attempts$ from$ $5$ (username$ attempted:$ $1)
!
! Once a successful AUTH occurs, remove the entry for that username in the
! bad_password_attempts table, and remove the entry for that source IP in
! the bad_user_attempts table. We want to try to remove the source IP entry in
```

```
! the second table, even if (for some reason) the MeterMaid callout on the
! first fails, so we split the store calls into two separate entries, rather
! than attempting two routine calls from one right hand side.
!
tcp_*|+|U|*|*|*authentication$ successful*|TCP|$_*|$_*|$_*|* $CCLEAR|$2|$7
!
! If the authentication was successful, then the probe is now:
! CLEAR|username|source-ip
!
CLEAR|*|* \
$CCLEAR|$0|$1$[IMTA_LIB:check_metermaid.so,remove,bad_password_attempts,$0]
CLEAR|*|* $[IMTA_LIB:check_metermaid.so,remove,bad_user_attempts,$1]
```

In the above example, it is convenient to detect successful authentication via the diagnostics field (the SMTP response). However, as of Messaging Server 7.3-11.01, an equivalent approach would be to detect successful authentication via the reason field (and use `log_reason=3`): as of Messaging Server 7.3-11.01, the reason field in cases of successful authentication would either be "authentication successful" or "authentication successful - switched to channel channel-name". (In prior versions, the reason field confounded some cases.)

55.1.2.3.5 Syslog notices when time-in-queue becomes "high", ceasing after any quick delivery

(Note that this example uses the MeterMaid "remove" routine, new in Messaging Server 7.2-0.01.) Another example of generating syslog notices when some condition occurs, and then resetting the MeterMaid table entry (and hence stopping the syslog notices) when the condition ceases, would be for cases where messages, while getting delivered upon first delivery attempt, are not getting delivered sufficiently promptly for the site's taste. That is, considering only messages that actually do manage to get delivered upon first attempt rather than failing a delivery attempt and having to be retried later (hence inherently taking a relatively "long" time to be delivered), the site wishes to watch for cases where that initial, successful, delivery attempt nevertheless was rather "slow". Here we will record in MeterMaid the destination domain for each new ZZ* message whose delivery is "slow" (taking 300 seconds or more), and generate a syslog notice for such domains once 5 messages have been slow within an hour (3600 seconds), but reset (to 0) the MeterMaid table entry for that domain once a "quick delivery" (within 60 seconds) has occurred.

```
LOG_FILENAME=3
LOG_QUEUE_TIME=3
!
! Set SNDOPR_PRIORITY to a syslog facility+severity value that will
! coordinate with your syslog.conf configuration, e.g.
! SNDOPR_PRIORITY=20 to choose LOG_MAIL+LOG_WARNING syslog.conf handling.
!
SNDOPR_PRIORITY=20
```

```
metermaid.table.slow_delivery.data_type: string
metermaid.table.slow_delivery.max_entries: 2000
metermaid.table.slow_delivery.options: nocase
metermaid.table.slow_delivery.type: throttle
metermaid.table.slow_delivery.quota: 5
metermaid.table.slow_delivery.quota_time: 3600
```


Triggering effects from transaction logging with LOG_ACTION

```
! Bit 1/value 2 is not set in any of:
! LOG_NOTARY, LOG_FILENAME, LOG_ENVELOPE_ID, or LOG_MESSAGE_ID
LOG_REASON=3
LOG_USERNAME=3
LOG_DIAGNOSTICS=3
```

or in Unified Configuration:

```
msconfig> show log_*
role.mta.log_diagnostics = 3
role.mta.log_reason = 3
role.mta.log_username = 3
...and likely additional log_* MTA options settings...
```

and configutil MeterMaid table settings that, in addition to basic configuration not shown here, include:

```
metermaid.table.sends_to_bogus_recipients.data_type: string
metermaid.table.sends_to_bogus_recipients.max_entries: 5000
metermaid.table.sends_to_bogus_recipients.options: nocase,penalize
metermaid.table.sends_to_bogus_recipients.type: throttle
metermaid.table.sends_to_bogus_recipients.quota: 15
metermaid.table.sends_to_bogus_recipients.quota_time: 86400
```

then a LOG_ACTION mapping table to track in MeterMaid bad recipient addresses discovered at dequeue time, and a corresponding FROM_ACCESS mapping table that checks that MeterMaid data to decide whether to allow new message submissions, could be:

LOG_ACTION

```
! source-chan|dest-chan|action|size|env-from|orig-env-to|env-to|
! username|reason|diagnostics
!
tcp_local||R*|*|*|*|*|$**|Remote$ SMTP$ server$ has$ rejected$ address|* \
$[IMTA_LIB:check_metermaid.so,throttle,sends_to_bogus_recipients,$5]
tcp_local||K*|*|*|*|*|$**|Remote$ SMTP$ server$ has$ rejected$ address|* \
$[IMTA_LIB:check_metermaid.so,throttle,sends_to_bogus_recipients,$5]
```

FROM_ACCESS

```
TCP|*|*|*|*|SMTP*|MAIL|tcp_auth|*|* \
$[IMTA_LIB:check_metermaid.so,test,sends_to_bogus_recipients,$6,>=15]$NYou$ \
have$ sent$ to$ too$ many$ bad$ addresses$ today
```

Note that since in the logging (MTA transaction log entries and hence the LOG_ACTION field) the username field resulting from SMTP AUTH authentication is actually the mail attribute value with the asterisk character prefixed, whereas in FROM_ACCESS the "username" field is simply the pure mail attribute value, above in LOG_ACTION the entries make sure to explicitly match the asterisk character and then *not* include it in the sends_to_bogus_recipients table updates, so that the FROM_ACCESS probes can match on just the username.

55.1.2.3.7 Blocking dictionary attack on user name space (botnet attack)

Automated spam engines may mount a so-called "dictionary attack", attempting to run quickly through possible recipient addresses when submitting messages. So a source that frequently submits messages with many bad (invalid) addresses in the local domain may be regarded with suspicion. One possible use of [MeterMaid](#) with LOG_ACTION is to throttle submissions from senders who cause many "J" records ([rejections of attempted submissions](#)).

So with `deferralrejectlimit 4` set on the `tcp_local` channel, and with MTA options settings that include:

```
LOG_CONNECTION=515
LOG_DIAGNOSTICS=3
```

or in Unified Configuration

```
msconfig> show log_*
role.mta.log_connection = 515
role.mta.log_diagnostics = 3
...and likely additional log_* MTA options settings...
```

and `configutil` MeterMaid table settings that, in addition to basic configuration not shown here, includes configuration of a MeterMaid "J-by-IP" table and a "J-jail" table:

```
metermaid.table.J-by-IP.data_type: ipv4
metermaid.table.J-by-IP.max_entries: 10000
metermaid.table.J-by-IP.quota: 4
metermaid.table.J-by-IP.quota_time: 600
metermaid.table.J-jail.data_type: ipv4
metermaid.table.J-jail.max_entries: 10000
metermaid.table.J-jail.quota: 1
metermaid.table.J-jail.quota_time: 3600
```

or in Unified Configuration:

```
metermaid.table.J-by-IP.data_type: ipv4
metermaid.table.J-by-IP.max_entries: 10000
metermaid.table.J-by-IP.quota: 4
metermaid.table.J-by-IP.quota_time: 600
metermaid.table.J-jail.data_type: ipv4
metermaid.table.J-jail.max_entries: 10000
metermaid.table.J-jail.quota: 1
metermaid.table.J-jail.quota_time: 3600
```

then a LOG_ACTION mapping table to track in MeterMaid "J" records, and a corresponding [PORT_ACCESS mapping table](#) that checks that MeterMaid data to decide whether or not to allow connections, could be:

```
PORT_ACCESS
*|*|*|*|*  $C$|INTERNAL_IP;$3|$Y$E
```

```

TCP|*|*|*|*|* \
$:A$C$[IMTA_LIB:check_metermaid.so,test,J-jail,$2,>0]$N$2-blocked2$E
* $YEXTERNAL

DONT_JAIL

$<10.0.0.0/24> $N
* $Y

LOG_ACTION

! To count only the "J" records that exceeded the chosen toleration
! level of 3, we look for the deferralrejectlimit error text:
!
tcp_local|*|J*|*|rfc822;|*|451$ 4.5.3$ Too$ many$ rejections;$ \
try$ again$ later.*|TCP|*|*|*|*|SMTP \
$C$|DONT_JAIL;$6|\
$[IMTA_LIB:check_metermaid.so,throttle,j-by-ip,$6]\
$[IMTA_LIB:check_metermaid.so,throttle,j-jail,$6]$E

```

The asterisk at the end of the diagnostic text is so that even a "final" "J" record for an SMTP session -- as when the TCP/IP-channel-specific option [MAX_J_ENTRIES](#) is exceeded, resulting in some additional, extra text -- will be counted.

55.2 MTA counters

The MTA has facilities to collect and monitor channel counters based upon the Mail Monitoring MIB, [RFC 2789](#) (which updates [RFC 1566](#)). Note that counters are intended for providing real-time "snap-shots" of MTA behavior, rather than for gathering the sort of statistics instead available from the [MTA transaction log files](#). For a description of the MTA channel counters, see [MTA channel counters](#).

New in 8.0, the MTA also maintains eight signed, 64 bit counters intended for updating from within Sieve scripts, for site-customizable use; see the [Sieve adjustcounter extension](#).

The MTA provides [utilities](#) to display the counters directly; see the `imsimta counters` and `imsimta qm` utilities.

55.2.1 MTA channel counters

The MTA has facilities to collect and monitor channel counters based upon the Mail Monitoring MIB, [RFC 2789](#) (which updates [RFC 1566](#)). These counters tabulate on a per channel basis the twelve items described in [Table of channel counters from MADMAN MIB](#).

The MTA also supports some additional channel, association, and arbitrary counters; see this more extensive list at [Table of MTA counters](#).

Table 55.4 Channel counters from MADMAN MIB

Field name	Description
RECEIVED_MESSAGES	The number of messages enqueued to the channel
SUBMITTED_MESSAGES	The number of messages enqueued by the channel

STORED_MESSAGES	The total number of messages currently stored for the channel
DELIVERED_MESSAGES	The number of messages dequeued by the channel
RECEIVED_VOLUME	The volume of messages enqueued to the channel as measured in MTA blocks
SUBMITTED_VOLUME	The volume of messages enqueued by the channel as measured in MTA blocks
STORED_VOLUME	The volume of messages currently stored for the channel as measured in MTA blocks
DELIVERED_VOLUME	The volume of messages dequeued by the channel as measured in MTA blocks
RECEIVED_RECIPIENTS	The total number of recipients specified in all messages enqueued to the channel
SUBMITTED_RECIPIENTS	The total number of recipients specified in all messages enqueued by the channel
STORED_RECIPIENTS	The total number of recipients specified in all messages currently stored for the channel
DELIVERED_RECIPIENTS	The total number of recipients specified in all messages dequeued by the channel

An MTA block is, by default, 1024 bytes. However, this size may vary from system to system. The size of an MTA block is controlled with the [block_size](#) MTA option.

Table 55.5 MTA counters

Accessible via PMDF API	
Field name	Description
received_messages	The number of messages enqueued to the channel
stored_messages	The total number of messages currently stored for the channel
delivered_messages	The number of messages dequeued by the channel
submitted_messages	The number of messages enqueued by the channel
attempted_messages	The number of temporary errors encountered during attempted message dequeues by the channel
rejected_messages	The number of attempted messages enqueues rejected by the channel
failed_messages	The number of permanent errors (hard delivery failures) encountered during attempted messages dequeues by the channel
received_volume	The volume of messages enqueued to the channel as measured in MTA blocks
stored_volume	The volume of messages currently stored for the channel as measured in MTA blocks
delivered_volume	The volume of messages dequeued by the channel as measured in MTA blocks
submitted_volume	The volume of messages enqueued by the channel as measured in MTA blocks
attempted_volume	The volume of messages that encountered temporary errors during attempted message dequeues by the channel as measured in MTA blocks

rejected_volume	The volume of the messages for message enqueues rejected by the channel as measured in MTA blocks
failed_volume	The number of permanent errors (hard delivery failures) encountered during attempted messages dequeues by the channel as measured in MTA blocks
received_recipients	The total number of recipients specified in all messages enqueued to the channel
stored_recipients	The total number of recipients specified in all messages currently stored for the channel
delivered_recipients	The total number of recipients specified in all messages dequeued by the channel
submitted_recipients	The total number of recipients specified in all messages enqueued by the channel
attempted_recipients	The total number of recipients encountering temporary errors during attempted message dequeues by the channel
rejected_recipients	The total number of recipients rejected by the channel
failed_recipients	The total number of recipients encountering permanent errors (hard delivery failures) during attempted messages dequeues by the channel
delivered_first_messages	The total number of messages enqueued to the channel which were either successfully delivered, or returned as undeliverable, upon their first processing attempt
delivered_first_queue_count	Cumulative count of first message delivery attempts made by the channel. When this value is less than received_messages, it means that delivery has not yet been attempted for all received messages. This is not unusual: this value is expected to lag behind received_messages.
delivered_first_queue_time	Cumulative count of elapsed seconds between when a message is enqueued and when processing of its first delivery attempt completes. The result of dividing delivered_first_queue_time by delivered_first_queue_count gives the average amount of time in seconds spent by a message in the processing queues as it awaits its initial delivery attempt.
delivered_queue_count	Cumulative count of message delivery attempts made by the channel
delivered_queue_time	Cumulative count of elapsed seconds between when a message is enqueued and when it is finally removed from the channel queue. The result of dividing delivered_queue_time by delivered_queue_count gives the average amount of time in seconds spent by a message in the processing queues.
Association counters displayed by counters utility	
Field name	Description
accum_inbound_assocs	
current_inbound_assocs	
rejected_inbound_assocs	
failed_inbound_assocs	
accum_outbound_assocs	
current_outbound_assocs	

rejected_outBound_assocs	
failed_outbound_assocs	
Sieve adjustcounter counters	
Field name	Description
Sieve counter [1]	
Sieve counter [2]	
Sieve counter [3]	
Sieve counter [4]	
Sieve counter [5]	
Sieve counter [6]	
Sieve counter [7]	
Sieve counter [8]	
Additional message counters	
Field name	Description
DeliveredVolumeBins	
DeliveredFirstQueueCountBins	
DeliveredFirstQueueTimeBins	
DeliveredQueueCountBins	
DeliveredQueueTimeBins	
LastInbound	Most recent "E" or "J" record for this (source) channel
LastOutbound	Most recent "D", "Q", "Z", or "R" record for this (destination) channel; (Bug # 6774400 is that "K" records don't update this counter)
ConvSucceeded	For the conversion channel , identical to DeliveredMessages; 0 for all other channels
ConvFailed	For the conversion channel , identical to FailedMessages; 0 for all other channels
HeldCount	Obtained by periodically scanning the queue directories and counting .HELD files. Whether this scan is performed is controlled by the directoryscan SNMP option (in legacy configuration, the <code>local.snmp.directoryscanconfigutil</code> parameter) which defaults to 1---TRUE.
OldestAge	
OldestMsgid	

- 1. An MTA block is, by default, 1024 bytes. However, this size may vary from system to system. The size of an MTA block is controlled with the [block_size](#) MTA option.
- 2. rejected_volume does not generally capture the entire volume of the messages rejected by the channel. Depending upon when, during attempted message submission, the rejection occurs, little to none of the message content and size may have been transferred (or made known) to the channel.

It is important to note that these counters generally need to be looked at over time noting the minimum values seen. The minimums may actually be negative for some channels. Such a

negative value merely means that there were messages queued for a channel at the time that its counters were zeroed (*e.g.*, the cluster-wide database of counters created). When those messages were dequeued, the associated counters for the channel were decremented 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.

55.2.2 Purpose and use of MTA counters

MTA channel counters are intended for indicating the *health and performance trends* of your e-mail system. MTA channel counters are *neither designed nor intended* to provide an accurate accounting of message traffic; for precise accounting, instead see [MTA transaction logging](#). The lack of accuracy in the MTA's channel counters is an inherent aspect of their design; it is not a bug. Specifically, the MTA's channel counters adhere to what Marshall Rose calls the fundamental axiom of management, which is that management must itself not interfere with proper system and network operation by consuming anything but the tiniest amount of resource.

Therefore the MTA's channel counters are implemented using the lightest weight mechanisms available, namely a shared memory section on each system. 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. [MTA counters implementation](#) provides further discussion of the implementation of counters.

55.2.3 MTA counters implementation

For performance reasons, each MTA host keeps a cache of MTA channel counters in memory using a shared memory section (on UNIX). As processes on the host 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 `imsimta startup` command also creates the in-memory section, if it does not exist.)

The command `imsimta counters -show` or the `imsimta qm` command `counters show` may be used to show the values of the counters.

The command `imsimta counters -clear` or the `imsimta qm` command `counters clear` may be used to reset the counters to zero.

In addition to the above commands for directly displaying the MTA's counters, on some platforms an [SNMP subagent](#) may be available to serve the MTA counters out through SNMP to any standards-based SNMP monitoring station.

55.2.4 SNMP subagents

On some platforms, SNMP subagents are available to serve out the [MTA channel counters](#) using the Mail and Directory Management (MADMAN) SNMP MIB described in [RFC 2788](#) and [RFC 2789](#) (originally [RFC 1565](#) and [RFC 1566](#)). (See [MIB variables served](#) for a more detailed list of the exact MIB variables served, with OIDs and syntaces.) Presently, SNMP subagents are available for use with Net-SNMP used on Solaris platforms running Solaris 10, as well as Linux platforms.

Several options affect operation of these subagents; see [SNMP options](#).

55.2.4.1 MIB variables served

The [SNMP subagents](#) serve out selected variables from the MADMAN MIBs (see [RFC 2788](#) and [RFC 2789](#), updating [RFC 1565](#) and [1566](#)), specifically, those variables from the `applicationTable`, `mtaTable`, and `mtaGroupTable` tables as shown in [Table of supported MIB variables](#).

Table 55.6 Supported MIB variables

applicationTable variables			
Variable name	OID	Syntax	Value
<code>applName</code>	mib-2.27.1.1.2	SnmpAdminString	<i>instance-name</i> <i>service-name</i> <i>on host-name</i>
<code>applVersion</code>	mib-2.27.1.1.4	SnmpAdminString	<i>major-version</i> . <i>minor-version</i> or <i>major-version</i> . <i>minor-version</i> <i>Patch</i> <i>patch-version</i>
<code>applDirectoryName</code>	mib-2.27.1.1.3	SnmpAdminString	NULL
<code>applUpTime</code>	mib-2.27.1.1.5	TimeStamp	If the Job Controller is running, then the time (in hundredths of seconds) since the Job Controller was started up (or restarted); otherwise, 0
<code>applOperStatus</code>	mib-2.27.1.1.6	Integer	1 (up) or 2 (down)
<code>applLastChange</code>	mib-2.27.1.1.7	TimeStamp	Time (in hundredths of seconds) since the Job Controller's pidfile was modified
<code>applInboundAssociations</code>	mib-2.27.1.1.8	Gauge32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupInboundAssociations</code>
<code>applOutboundAssociations</code>	mib-2.27.1.1.9	Gauge32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupOutboundAssociations</code>
<code>applAccumulatedInboundAssociations</code>	mib-2.27.1.1.10	Counter32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupAccumulatedInboundAssociations</code>
<code>applAccumulatedOutboundAssociations</code>	mib-2.27.1.1.11	Counter32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupAccumulatedOutboundAssociations</code>
<code>applLastInboundActivity</code>	mib-2.27.1.1.12	TimeStamp	Most recent of the <code>mtaGroupLastInboundActivity</code> values
<code>applLastOutboundActivity</code>	mib-2.27.1.1.13	TimeStamp	Most recent of the <code>mtaGroupLastInboundActivity</code> values
<code>applRejectedInboundAssociations</code>	mib-2.27.1.1.14	Counter32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupRejectedInboundAssociations</code>
<code>applFailedOutboundAssociations</code>	mib-2.27.1.1.15	Counter32	Sum over all <code>mtaGroups</code> (channels) of the <code>mtaGroupFailedOutboundAssociations</code>
<code>applDescription</code>	mib-2.27.1.1.16	SnmpAdminString	<i>ims-name</i> <i>applVersion-value</i>
<code>applURL</code>	mib-2.27.1.1.17	URLString	NULL
mtaTable variables			
Variable name	OID	Syntax	Derived from MTA counter
<code>mtaReceivedMessages</code>	mib-2.28.1.1.1	Counter32	<code>received_messages</code>
<code>mtaStoredMessages</code>	mib-2.28.1.1.2	Gauge32	<code>stored_messages</code>
<code>mtaTransmittedMessages</code>	mib-2.28.1.1.3	Counter32	<code>delivered_messages</code>
<code>mtaReceivedVolume</code>	mib-2.28.1.1.4	Counter32	<code>received_volume</code> (converted, as necessary, to Kbytes)
<code>mtaStoredVolume</code>	mib-2.28.1.1.5	Gauge32	<code>stored_volume</code> (converted, as necessary, to Kbytes)
<code>mtaTransmittedVolume</code>	mib-2.28.1.1.6	Counter32	<code>delivered_volume</code> (converted, as necessary, to Kbytes)
<code>mtaReceivedRecipients</code>	mib-2.28.1.1.7	Counter32	<code>received_recipients</code>
<code>mtaStoredRecipients</code>	mib-2.28.1.1.8	Gauge32	<code>stored_recipients</code>
<code>mtaTransmittedRecipients</code>	mib-2.28.1.1.9	Counter32	<code>delivered_recipients</code>
<code>mtaSuccessfulConvertedMessages</code>	mib-2.28.1.1.10	Counter32	Sum over all <code>mtaGroups</code> (channels) of <code>mtaGroupSuccessfulConvertedMessages</code> ; since all <code>non-conversion*</code> channels have

SNMP subagents

			value 0, this ends up being the sum over just conversion* channels
mtaFailedConvertedMessages	mib-2.28.1.1.11	Counter32	Sum over all mtaGroups (channels) of mtaGroupFailedConvertedMessages; since all non- conversion* channels have value 0, this ends up being the sum over just conversion* channels
mtaLoopsDetected	mib-2.28.1.1.12	Counter32	Sum over all mtaGroups (channels) of mtaGroupLoopsDetected
mtaGroupTable variables			
Variable name	OID	Syntax	Derived from MTA counter
mtaGroupReceivedMessages	mib-2.28.2.1.2	Counter32	received_messages
mtaGroupRejectedMessages	mib-2.28.2.1.3	Counter32	rejected_messages
mtaGroupStoredMessages	mib-2.28.2.1.4	Gauge32	stored_messages
mtaGroupTransmittedMessages	mib-2.28.2.1.5	Counter32	delivered_messages
mtaGroupReceivedVolume	mib-2.28.2.1.6	Counter32	received_volume (converted, as necessary, to Kbytes)
mtaGroupStoredVolume	mib-2.28.2.1.7	Gauge32	stored_volume (converted, as necessary, to Kbytes)
mtaGroupTransmittedVolume	mib-2.28.2.1.8	Counter32	delivered_volume (converted, as necessary, to Kbytes)
mtaGroupReceivedRecipients	mib-2.28.2.1.9	Counter32	received_recipients
mtaGroupStoredRecipients	mib-2.28.2.1.10	Gauge32	stored_recipients
mtaGroupTransmittedRecipients	mib-2.28.2.1.11	Counter32	delivered_recipients
mtaGroupName	mib-2.28.2.1.25	String	channel_name
mtaGroupInboundAssociations	mib-2.28.2.1.13	Gauge32	current_inbound_assocs
mtaGroupOutboundAssociations	mib-2.28.2.1.14	Gauge32	current_outbound_assocs
mtaGroupAccumulatedInboundAssociations	mib-2.28.2.1.15	Counter32	accum_inbound_assocs
mtaGroupAccumulatedOutboundAssociations	mib-2.28.2.1.16	Counter32	accum_outbound_assocs
mtaGroupRejectedInboundAssociations	mib-2.28.2.1.19	Counter32	rejected_inbound_assocs
mtaGroupFailedOutboundAssociations	mib-2.28.2.1.20	Counter32	failed_outbound_assocs
mtaGroupOldestMessageStored	mib-2.28.2.1.12	TimeInterval	oldest_age: Time, in hundredths of a second, that the oldest, non-.HELD message in the channel queue has been present; the Job Controller maintains this information and updates the oldest_age MTA counter
mtaGroupLastInboundActivity	mib-2.28.2.1.17	TimeInterval	Channel counter lastinbound
mtaGroupLastOutboundActivity	mib-2.28.2.1.18	TimeInterval	Channel counter lastoutbound
mtaGroupInboundRejectionReason	mib-2.28.2.1.21	SnmpAdminString	NULL
mtaGroupOutboundConnectFailureReason	mib-2.28.2.1.22	SnmpAdminString	NULL
mtaGroupScheduledRetry	mib-2.28.2.1.23	TimeInterval	0
mtaGroupMailProtocol	mib-2.28.2.1.24	OID	<i>smtp-oid, i.e., 9.1.3.6.1.2.1.27.4.25</i>
mtaGroupSuccessfulConvertedMessages	mib-2.28.2.1.26	Counter32	delivered_messages for a conversion* channel ; 0 for all other channels
mtaGroupFailedConvertedMessages	mib-2.28.2.1.27	Counter32	failed_messages for a conversion* channel ; 0 for all other channels
mtaGroupDescription	mib-2.28.2.1.28	SnmpAdminString	<i>instance-name MTA channel-name channel</i>
mtaGroupURL	mib-2.28.2.1.29	URLString	NULL
mtaGroupCreationTime	mib-2.28.2.1.30	TimeInterval	0x7FFFFFFF
mtaGroupHierarchy	mib-2.28.2.1.31	Integer	0
mtaGroupOldestMessageId	mib-2.28.2.1.32	SnmpAdminString	oldest_msgid: Message-id of the oldest, non-.HELD message present in the channel

			queue; the Job Controller updates the oldest_msgid MTA counter
mtaGroupLoopsDetected	mib-2.28.2.1.33	Counter32	Corresponds to the MTA's private HeldCount counter (which is a cached count of .HELD files seen during a periodic scan of the channel queue directory)
mtaGroupLastOutboundAssociationAttempt	mib-2.28.2.1.34	TimeInterval	mtaGroupLastOutboundActivity value

Note: the OID for mib-2 is 1.3.6.1.2.1.

Each [MTA channel](#) is identified with with an MTA group. Thus, for each channel, there will be a row in the mtaGroupTable. For example, if there are M channels, the OID mib-2.28.2.1.25. n gives the name of the channel associated with the n th row in the table where n satisfies $1 \leq n \leq M$.

Only one application and MTA is recognized by the [subagent](#) and consequently there is only one row in the applicationTable and mtaTable tables. The only valid instance identifier for those two tables is thus ".1"; *i.e.*, for either table, the OID for an instance of a variable is formed by taking the OID of the variable and appending ".1" to it. For example, a `get` operation on mib-2.27.1.1.4.1 would return the version number of MTA.

Each row of the mtaGroupTable table corresponds to a set of [MTA channel counters](#) maintained by the MTA. A description of each mtaGroupTable variable is given in [Table of mtaGroupTable variable descriptions](#). These counters may be directly manipulated `imsimta counters` utility or the `imsimta qm counters` utility's `counters` command. Refer to the discussion of [MTA channel counters](#) for further information on the MTA channel counters.

Table 55.7 mtaGroupTable variable descriptions

mtaGroupTable variable	MTA counter	Description
mtaGroupReceivedMessages	RECEIVED_MESSAGES	Count of messages enqueued to the channel.
mtaGroupStoredMessages	STORED_MESSAGES	Count of messages enqueued to the channel but not yet delivered.
mtaGroupTransmittedMessages	DELIVERED_MESSAGES	Count of messages delivered (dequeued) by the channel.
mtaGroupReceivedVolume	RECEIVED_VOLUME	Volume of messages enqueued to the channel as measured in Kbytes = 1024 bytes.
mtaGroupStoredVolume	STORED_VOLUME	Volume of messages enqueued to the channel but not yet delivered as measured in Kbytes.
mtaGroupTransmittedVolume	DELIVERED_VOLUME	Volume of messages which have been delivered (dequeued) by the channel as measured in Kbytes.
mtaGroupReceivedRecipients	RECEIVED_RECIPIENTS	Volume of messages enqueued to the channel as measured by the total number of envelope recipient addresses.

mtaGroupStoredRecipients	STORED_RECIPIENTS	Volume of messages enqueued to the channel but not yet delivered as measured by the total number of envelope recipient addresses.
mtaGroupTransmittedRecipients	DELIVERED_RECIPIENTS	Volume of messages which have been delivered (dequeued) by the channel as measured by the total number of envelope recipient addresses.
mtaGroupName		Name of the channel.

The values in the mtaTable correspond to the column sums of the mtaGroupTable; *e.g.*, mtaReceivedMessages is the sum over all rows of the mtaGroupTable column mtaGroupReceivedMessages.

Note: The underlying [MTA channel counters](#) may take on negative values. However, the corresponding MIB variables must be non-negative. To reconcile this difference, the [subagent](#) tracks the minimum value seen for each channel counter and then uses that minimum to adjust the MIB variable such that it has a minimum of zero. This is done by subtracting the minimum value from the counter when that minimum is less than zero. For this reason, the values of the counters displayed with the `imsimta counters` command may differ from those displayed from an SNMP client.

Note: The MIB volume variables measure in units of kilobytes, whereas MTA counters measure message volume in units of [MTA blocks](#). While the default MTA block size (`block_size` MTA option) is 1024 bytes = 1 kilobyte and thus identical to the units for the MIB volume variables, if the MTA block size has been set to some other value, then the MTA counters for volumes will be in different units than the kilobytes of MIB volume variables. The subagent will adjust the MTA counters volume values, if needed, to obtain kilobyte values for the MIB volume variables.

Chapter 56 MTA performance tuning

56.1 MTA performance: CPU and resources	56-1
56.2 MTA performance: Disks and files	56-3
56.3 System parameters on Solaris	56-3
56.3.1 For the Dispatcher:	56-3
56.3.2 For the Job Controller:	56-4

There are a variety of things which may be done to improve the MTA's performance. However, before trying to tune the MTA you should first feel comfortable with the MTA: have a basic understanding of how it works, be familiar with your configuration, and be able to recognize when the MTA isn't working on your system. In addition, it is important that you spend some time identifying what the bottlenecks are on your system: CPU resources, disk speed, memory, network speed or latencies, etc. Without a clear idea of where the bottlenecks are, any tuning you do is likely to be ineffective.

56.1 MTA performance: CPU and resources

Not necessarily in order of importance, here are some points to consider:

- In versions past, it was important to use a [compiled configuration](#), to reduce the startup time of MTA processing jobs. As of 7.0.5 and Unified Configuration, this is no longer important.
- If your system has the memory to spare, increasing the size of message that processing jobs can buffer internally can reduce use of temporary buffer files on disk when receiving or processing large messages. See the discussion of the [max_internal_blocks MTA option](#).
- Ensure that you have sufficient swap space for the needs of your configuration.
- Consider [Job Controller pools](#) for specific channels which you wish to ensure always have processing slots. The usual initial configuration establishes a basic set of pools suitable for the typical use of the channels in the initial configuration, but if you have added special purpose channels to your configuration, or if your site's message traffic has special characteristics or you have special needs, then you may benefit from adding new pool definitions and/or assigning particular channels to different pools. For instance, if you have a dedicated, heavy-use TCP/IP channel for sending to some sister site, then you may wish to define a separate pool that will be dedicated for the use of that special channel. Then use the [pool channel option](#) to direct the special channel to run in that new pool.
- The [Dispatcher](#) controls the creation and use of multithreaded SMTP server processes. If, as is typical, incoming SMTP over TCP/IP messages are a major component of e-mail traffic at your site, monitor how many simultaneous incoming SMTP connections you tend to have, and the pacing at which such connections come in. Tuning of [Dispatcher configuration options](#) controlling the number of SMTP server processes, the number of connections each can handle, the threshold at which new server processes are created, etc., may be beneficial if your site's incoming SMTP over TCP/IP traffic is unusually high or low.
- In general, for outgoing SMTP over TCP/IP channels, adjustment of the number of messages handled per thread ([threaddepth channel option](#)), the maximum number of threads per process ([MAX_CLIENT_THREADS](#) TCP/IP-channel-specific option), the maximum number of processes for the channel ([maxjobs channel option](#), potentially also limited

by the Job Controller's `job_limit` option value for the `pool` in which the channel runs), can potentially be beneficial, especially if your site's message traffic has out of the ordinary characteristics. For typical SMTP over TCP/IP channels, used to send to multiple different remote systems, the MTA's multithreaded TCP/IP channel's default behavior of sorting messages to different destinations into different threads and then handling all messages (up to the channel's `threaddepth` channel option's value) to a single host in a single thread is desirable for performance; indeed, in some cases increasing such a channel's `threaddepth` may be useful. However, for a `daemon` TCP/IP channel, one dedicated to sending to a specific system, if the receiving system supports multiple simultaneous connections it may be preferable to force the MTA to split the outgoing messages into separate threads at a lower threshold, by using a "small" `threaddepth` value, and then perhaps correspondingly adjusting to allow more threads and/or more processes using the `MAX_CLIENT_THREADS` channel option or the `maxjobs channel option`, respectively. And see also the discussion above regarding defining and using new pools for particular channels, to control the degree of resource sharing/resource competition among separate outgoing TCP/IP channels.

- Disabling the MTA's SMTP server creation and use of *.data-failed files by setting the TCP/I-channel-specific option `REUSE_TIMED_OUT_TRANSFERS=0` can provide a noticeable performance (throughput) increase---perhaps 30% for the SMTP server; the downside is that disabling their use means that the MTA will no longer detect and avoid certain cases of duplicate submissions of messages, so users may receive "duplicate" copies of messages that might have been avoided. (The main performance issue here is not usually the creation of the actual files, but rather the CPU used while deciding whether to make such a file; that is, the performance impact of such potential file use exists whether or not any *.data-failed file is in fact ever created.)
- Another channel heavily used at typical sites is the `ims-ms channel` for delivery to the Message Store. This, like the TCP/IP channels, is a multithreaded channel, and controls over its `threaddepth`, `DELIVER_THREADS` `ims-ms-channel-specific option`, `maxjobs`, and the `job_limit` for the `pool` in which it runs can potentially be relevant to performance, particularly if your site's needs are out of the ordinary.
- In `direct LDAP mode`, the caching of LDAP lookups represents a tradeoff between efficiency ("large" caches), *vs.* memory usage (which should not be too great a burden on a reasonably sized system) and small latency in LDAP entry changes ("small" caches). See especially the `alias_entry_cache_size`, `alias_entry_cache_timeout`, `reverse_address_cache_size`, `reverse_address_cache_timeout`, and `ldap_domain_timeout` MTA options, and to a lesser degree the `domain_match_cache_size` and `domain_match_cache_timeout` MTA options, which all fall into the category of `LDAP lookup caching`. Sites using `Sieve filters` should also see `filter_cache_size` and `filter_cache_timeout`; sites making heavy use of custom LDAP callouts (from mapping tables or rewrite rules) should also see `url_result_cache_size` and `url_result_cache_timeout`, also in that same category of `LDAP lookup caching`.
- When generating very large "mass mailings", see [Performance submitting mass mail messages](#).
- Note that increased processing or filtering of messages, such as that resulting from processing the inner levels of messages (`inner`, `innertrim` channel options), "sniffing" of message bodies (`thurman` or `uma` channel options or `CHARSET-CONVERSION` keywords), use of `character set conversion`, use of the `conversion channel`, use of `Sieve filters`, use of `callouts to third party virus or spam detection software such as Brightmail or Spamassassin`, MIME message fragmentation or defragmentation (`maxblocks`, `defragment` channel

options), use of TLS for encryption of SMTP messages (`*tls*` channel options), use of DNS reverse lookups for incoming SMTP messages (`ident*` channel options, `forwardcheck*` channel options, `mailfromdnsverify` channel option, or `dns_verify*` [image callouts](#) from an `*_ACCESS mapping table`), *etc.*, requires the MTA to do extra work hence of course has a performance impact. This is not to discourage use of such features---such features exist because they are useful, and desirable features for message processing---but do keep in mind the potential performance impact of increased message processing.

- In particular, especially for [system-wide](#) or channel [Sieve filters](#), it may be worth paying some attention to ensure that such Sieves use reasonable test logic, and are not really excessively long (excessively many tests).
- Also, when using mapping tables, such as `*_ACCESS mapping tables`, it is worthwhile to set up the mapping tables in an efficient way, without excessive numbers of entries, and attempting to keep the pattern matching reasonably efficient. As a rough rule of thumb, consider that once a mapping table has reached about 50 entries (lines), it is worth considering whether the table could be restructured to make use of [general database callouts](#) to do some of the specific matching, rather than having all matching text explicitly present in the mapping table itself.
- When [callouts to third party spam/virus filter packages](#) such as Brightmail or Spamassassin are part of the configuration, it is important to size and tune those third party filter packages adequately: their important filtering functionality tends to be inherently "expensive".

56.2 MTA performance: Disks and files

One of the more common bottlenecks for the MTA itself is disk I/O -- especially if the MTA is using the same disks or disk controller as the Message Store, which does a *lot* of disk I/O! The MTA itself does a lot of disk I/O. Try to keep the disks with the MTA's message queues below 66% capacity so that the operating system can efficiently manage file create and delete cycles. Also, use disk striping or other aggregate disk spindle techniques that help both read and writes. Avoid disk shadowing if possible. Disk is cheap these days: spend money on multiple spindles and sufficient free space.

More TBD

56.3 System parameters on Solaris

56.3.1 For the Dispatcher:

As of Solaris 11, the `sifp_fd_max` parameter in `/etc/system` is likely to need to be increased from its default of 50 (which may be much too small for the Dispatcher's needs in handing off file descriptors to its worker processes) to some much larger value -- perhaps 50,000.

The system's heap size (`datasize`) must be enough to accommodate the Dispatcher's thread stack usage. 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.

To display the heap size (*i.e.*, default `datasize` as reported by `limit` or default data size as reported by `ulimit`), use the `cs` command

```
# limit
```

or the ksh command (see the data result)

```
# ulimit -a
```

or the utility

```
# sysdef
```

56.3.2 For the Job Controller:

As of JES MS 6.4, the [Job Controller](#) will set its file descriptor limit to the lesser of 1024 or `rlim_fd_max`. Previously, the `rlim_fd_max` value would be used---which on systems with a high such value (in particular, on Solaris 10 where the default is 65535), would mean that the Job Controller -- and jobs it forks---could potentially spend significant time closing a great many unused file descriptors.

If running a version of the MTA prior to JES MS 6.4, and if the system has a high `rlim_fd_max` configured, it will benefit performance to modify the script that starts the Job Controller to issue an `rlimit` command setting a smaller file descriptor hard limit (for instance, 1024) before starting the Job Controller.

Chapter 57 Restricting information emitted

57.1 SMTP probe commands	57-1
57.2 Internal host names in Received: and Message-Id: header lines	57-2
57.3 Extra concerns for address canonicalization	57-4

Some sites will have special, heightened concerns regarding emitting information about their internal user names, host names, *etc.* There are various ways that information may "leak out"; some ways of controlling such information leakage are discussed here.

57.1 SMTP probe commands

During an SMTP connection, a remote sending side (or a person manually telnetting to your SMTP port) can issue commands requesting information such as a check on the validity of addresses. This very useful information can, however, be subject to abuse, *e.g.*, by automated search engines checking for valid email addresses on your firewall system. Therefore some sites may have an interest in disabling these helpful features.

Setting the option `DISABLE_EXPAND=1` for your Internet TCP/IP channel (typically `tcp_local`) disables the SMTP EXPN command. The SMTP EXPN command is normally used to expand (get the membership of) mailing lists. Note that the alias options `alias_expandable` and `alias_nonexpandable` (in legacy configuration, `mailing list named parameters [EXPANDABLE]` and `[NONEXPANDABLE]`) can be used on a per-mailing-list basis to control the SMTP EXPN response for that mailing list. Note also that use of mailing list access controls, *e.g.*, LDAP list attributes such as `mgrpAllowedBroadcaster` and `mgrpDisallowedBroadcaster` (or more precisely, those attributes named by the `ldap_auth_url` and `ldap_cant_url` MTA options), or `alias options alias_auth_list`, `alias_auth_mapping`, *etc.*, (corresponding in legacy configuration to `named parameters such as [AUTH_LIST], [AUTH_MAPPING], etc.`), also affect the SMTP EXPN response for the mailing list so marked: only if an SMTP client has passed the access controls (for instance by issuing a MAIL FROM: command identifying as a sender allowed to post to the list) will the MTA's SMTP server then return an informative response to the client's SMTP EXPN command. So `DISABLE_EXPAND=1` is suitable if you wish to disable *all* EXPN responses. However, if you only have some "sensitive" lists you can instead effectively get per-list controls on EXPN use.

Setting `HIDE_VERIFY=1` for your Internet TCP/IP channel causes the MTA to return a "generic" response to the SMTP VRFY command. The SMTP VRFY command is normally used to check whether an address is a legitimate address on the local system. (Note that as it is required that SMTP servers support the VRFY command, the MTA has to return some sort of response; with `HIDE_VERIFY=1`, this response is simply a "maybe" sort of response rather than an explicit yes or no.) See also `domainvrfy and related channel options` for a discussion of channel options that can also be used to affect SMTP VRFY responses.

Setting `DISABLE_ADDRESS=1` for your Internet TCP/IP channel causes the MTA to disable responses to its SMTP server's private XADR command, which normally returns information about the channel an address matches.

Setting `DISABLE_CIRCUIT=1` for your Internet TCP/IP channel causes the MTA to disable responses to the its SMTP server's private XCIR command, which normally returns information about the MTA message circuit checking facility.

Setting `DISABLE_STATUS=1` for your Internet TCP/IP channel causes the MTA to disable responses to its SMTP server's private XSTA command, which normally returns information about the numbers of messages in MTA queues.

Setting `DISABLE_GENERAL=1` for your Internet TCP/IP channel option file causes the MTA to disable responses to its SMTP server's private XGEN command, which normally returns status information about whether an MTA compiled configuration and character set are in use.

Sample `msconfig` commands to disable such probes on a typical `tcp_local` channel would be:

```
msconfig> set channel:tcp_local.options:DISABLE_EXPAND 1
msconfig# set channel:tcp_local.options:HIDE_VERIFY 1
msconfig# set channel:tcp_local.options:DISABLE_ADDRESS 1
msconfig# set channel:tcp_local.options:DISABLE_CIRCUIT 1
msconfig# set channel:tcp_local.options:DISABLE_STATUS 1
msconfig# set channel:tcp_local.options:DISABLE_GENERAL 1
```

For legacy configuration, a sample TCP/IP channel option file to disable probing via the SMTP server, for a site using a `tcp_local` channel, would be as shown below:

```
DISABLE_EXPAND=1
HIDE_VERIFY=1
DISABLE_ADDRESS=1
DISABLE_CIRCUIT=1
DISABLE_STATUS=1
DISABLE_GENERAL=1
```

See [TCPIP-channel-specific options](#) for more details on these options.

57.2 Internal host names in Received: and Message-Id: header lines

Received: headers are normally exceptionally useful headers for displaying the routing that a message really took. Their worth can be particularly apparent in cases of dealing with apparently forged email, or in cases where one is trying to track down what happened to a broken messages, or in cases where a message does not appear to be reliable and one is trying to figure out who might know how to respond to the message. Received: headers are also used by the Messaging Server MTA (and other mailers) to try to detect message loops.

Message-id: headers are normally useful for message tracking and correlation.

However, on the converse side, Received: headers on messages you send out give the message recipient information about the routing that a message really took through your internal systems and tend to include internal system names and possibly an envelope recipient address. And Message-id: headers tend to include internal system names. At some sites, this may be considered a security exposure.

If your site is concerned about this information being emitted, first see if you can configure your internal systems to control what information they put in these headers. For instance, the MTA options `received_domain` and `id_domain` can be used on a Messaging Server

system to specify the domain name to use when constructing Received: headers and Message-id: headers, respectively. Although these options are not usually particularly relevant on an edge MTA system -- an edge system is by definition a system whose name is intended to be visible to the outside world --- if you have the MTA on internal systems also, the options may be of interest on those internal MTA systems. In a similar spirit, the channel keyword [noreceivedfor](#) can be used on channels on an MTA system to instruct the MTA not to include the envelope recipient address in the Received: header it constructs, if limiting the exposure of internal "routing" addresses is a concern for your site. And for those rare cases where the inclusion of original envelope From information in Received: headers constructed is of concern, the channel keyword [noreceivedfrom](#) can be used on channels on an MTA system to instruct the MTA not to include envelope From information in Received: headers it constructs in those cases (involving changing the envelope From, such as certain sorts of mailing list expansions) where the MTA would normally include the envelope From: address.

Note that for messages that web clients submit through the MSHTTPD server to the MTA, the MSHTTPD server will (as of iMS 5.2p2) use any original client source IP present in an X-Forwarded-for: or Client-ip: or (as of JES MS 6.3) Proxy-ip: HTTP header to construct a "(Forwarded-for: *source-ip*)" comment clause in the Received: header line it (MSHTTP) submits to the MTA.

If necessary, address reversal on an MTA system can be used to "canonicalize" message id's, to remove undesired information, (though note that this removal of information may mean that the resulting message id's are no longer particularly useful). Note that the [use_reverse_database](#) MTA option must have bit 6 (value 64) set in order for address reversal to apply to message id's; for instance, if the option was previously set to the default value of 5, it must be set to 69 to apply to message id's. For instance, a site domain.com that wishes to ensure that no *host.domain.com* domains appear in message id's might use a [REVERSE mapping](#) such as:

REVERSE

```
*@*.domain.com      $C$:I$0@domain.com$Y$E
```

This REVERSE mapping only applies to message id's, due to the \$: I flag.

As regards Received: headers, only if you cannot configure your internal systems to control such sorts of information should you consider resorting to stripping such headers off entirely. Received: headers should not be removed lightly, due to their many and important uses, but if the internal routing and system name information in them is sensitive for your site and if you cannot configure your internal systems to control what information appears in these headers, then you may wish to strip off those headers on messages going out to the Internet via header trimming on your outgoing TCP/IP channel.

Note: Do **not** remove Received: headers or remove or simplify Message-id: headers on general principles or because your users do not like them! Removing such headers, among other things, (1) removes one of the best tracking mechanisms you have, (2) removes information that may be critical in tracking down and solving problems, (3) removes one of the few (and best) warnings of forged mail you may have, and (4) blocks the mail system's ability to detect and short-circuit message loops. Only remove such headers if you *know* your site *needs* them removed.

To implement header trimming, put the [headertrim](#) channel option --- you will probably want the [innertrim](#) channel option as well --- on your outgoing external TCP/IP channel or

channels, generally [tcp_local](#) and possibly other [tcp_*](#) channels (possibly every [tcp_*](#) channel except your internal channel, [tcp_internal](#)), and create a header trimming file for each such channel. The [headertrim](#) keyword causes header trimming to be applied to the outer message headers; the [innertrim](#) keyword causes the header trimming to be applied also to embedded message parts ([message/rfc822](#) parts) within the message. A sample header trimming file for a site using a [tcp_local](#) channel is shown below:

```
Received: MAXIMUM=-1
MR-Received: MAXIMUM=-1
X400-Received: MAXIMUM=-1
```

See the [headertrim](#) channel option for more details on header trimming.

As of JES MS 6.3, the [Sieve "deleteheader" and "replaceheader" actions](#) provide another approach -- very powerful, so use with caution! -- to modifying header lines to limit emission of internal host names.

57.3 Extra concerns for address canonicalization

Proper address canonicalization (converting any "internal" address forms to canonicalized, appropriate-for-external-use, forms) is, nowadays, typically part of proper domain provisioning in LDAP, in particular proper use of LDAP domain attributes such as (in Schema 1) [aliasedObjectName](#) and [inetCanonicalDomainName](#) and [aliasedObjectName](#) (or more precisely, those LDAP domain attributes named by the [ldap_domain_attr_canonical](#) and [ldap_domain_attr_alias](#) MTA options) or (in Schema 2) [sunPreferredDomain](#) and [associatedDomain](#) (or more precisely, those LDAP domain attributes named by the [ldap_attr_domain1_schema2](#) and [ldap_attr_domain2_schema2](#) MTA options) and proper user provisioning in LDAP, in particular proper use of [mail](#), [mailAlternateAddress](#), and [mailEquivalentAddress](#) LDAP attributes (or more precisely, those LDAP user attributes named by the [ldap_primary_address](#), [ldap_alias_addresses](#), and [ldap_equivalence_addresses](#) MTA options). However, there are a few additional configuration items that may be of interest.

1. Put the [inner](#) keyword on (at least) your channels outgoing to the external world so that address rewriting will be applied to address in embedded message parts ([message/rfc822](#) parts).
2. If you do not wish notification messages generated by MTA systems the internal address, then you may wish to use the [suppressfinal](#) channel option.

Chapter 58 MTA command line utilities

58.1	cache -change	58-3
58.1.1	Syntax	58-3
58.1.2	Restrictions	58-3
58.1.3	Parameters	58-3
58.1.4	Description	58-3
58.1.5	Switches	58-4
58.1.6	Examples	58-5
58.2	cache -synchronize	58-6
58.2.1	Syntax	58-6
58.2.2	Restrictions	58-6
58.2.3	Parameters	58-6
58.2.4	Description	58-6
58.2.5	Switches	58-6
58.2.6	Examples	58-6
58.3	cache -walk	58-8
58.3.1	Syntax	58-8
58.3.2	Restrictions	58-8
58.3.3	Parameters	58-8
58.3.4	Description	58-8
58.3.5	Switches	58-8
58.3.6	Examples	58-8
58.4	chbuild	58-9
58.4.1	Syntax	58-9
58.4.2	Restrictions	58-9
58.4.3	Parameters	58-9
58.4.4	Description	58-9
58.4.5	Switches	58-10
58.4.6	Examples	58-11
58.5	cnbuild	58-12
58.5.1	Syntax	58-12
58.5.2	Restrictions	58-12
58.5.3	Parameters	58-12
58.5.4	Description	58-12
58.5.5	Switches	58-16
58.5.6	Examples	58-18
58.6	crdb	58-19
58.6.1	Syntax	58-19
58.6.2	Restrictions	58-19
58.6.3	Parameters	58-19
58.6.4	Description	58-19
58.6.5	Switches	58-20
58.6.6	Examples	58-21
58.7	process	58-22
58.7.1	Syntax	58-22
58.7.2	Restrictions	58-22
58.7.3	Parameters	58-22
58.7.4	Description	58-22
58.7.5	Examples	58-22
58.8	purge	58-23
58.8.1	Syntax	58-23

58.8.2 Restrictions	58-23
58.8.3 Parameters	58-23
58.8.4 Description	58-23
58.8.5 Switches	58-23
58.8.6 Examples	58-24
58.9 reload	58-25
58.9.1 Syntax	58-25
58.9.2 Restrictions	58-25
58.9.3 Parameters	58-25
58.9.4 Description	58-25
58.10 test -match	58-26
58.10.1 Syntax	58-26
58.10.2 Restrictions	58-26
58.10.3 Parameters	58-26
58.10.4 Description	58-26
58.10.5 Examples	58-26
58.11 test -rewrite	58-28
58.11.1 Syntax	58-28
58.11.2 Parameters	58-30
58.11.3 Description	58-30
58.11.4 Switches	58-31
58.11.5 Examples	58-39
58.12 version	58-42
58.12.1 Syntax	58-42
58.12.2 restrictions	58-42
58.12.3 Parameters	58-42
58.12.4 Description	58-42
58.12.5 Example	58-42

The MTA contains a modest collection of management utility programs, which are used to perform various maintenance, testing, and management tasks.

These utilities are generally invoked via `imsimta` commands, *e.g.*,

```
# imsimta version
```

58.1 cache -change utility

Change [Job Controller option](#) effective values for the currently running [Job Controller](#) process.

58.1.1 Syntax

```
imsimta cache -change
```

Table 58.1 imsimta cache -change Command Switches

Switch	Default
-global	
-debug=n	
-max_messages=n	
-template=name	
-master_job=command	
-slave_job=command	
-channel=name	
-thread_depth=n	
-job_limit=n	
-parallel_rebuild=n	-parallel_rebuild=12
-inorder_rebuild	-noinorder_rebuild

58.1.2 Restrictions

Must have superuser privileges (UNIX) in order to use this utility.

58.1.3 Parameters

None.

58.1.4 Description

(New in JES MS 6.2.) The `imsimta cache -change` utility is used to change various [Job Controller option](#) effective values "on the fly" (that is, without requiring a [restart](#) of the Job Controller in order to take effect) for the currently running [Job Controller](#) process.

Exactly one of `-channel`, `-template`, or `-global` must be specified. `-channel` is for changing the behavior or definition of a *particular channel*: changing its effective `maxjobs` (`-job_limit`) or `threaddepth` (`-thread_depth`), or its `master_command` (`-master_job`) or `slave_command` (`-slave_job`). `-template` is for adding a new type of [channel](#), or for changing the definition of all channels of that type: (re)defining its `master_command` (`-master_job`) and/or `slave_command` (`-slave_job`). `-global` is for setting certain global Job Controller options: `debug`, `max_cache_messages` (`-max_messages`), or (6.2p2 and later?) `rebuild_parallel_channels` (`-parallel_rebuild`), or `rebuild_in_order` (`-inorder_rebuild`).

Note that [restarting](#) the Job Controller is very much to be avoided on a production MTA that has messages already in its queues. So for certain sorts of Job Controller configuration changes that one might want to make on a running MTA, especially configuration changes to trace down problems (debugging) or re-deploy resources at times of heavy load (change channel [maxjobs](#) via `-job_limit` or [threaddepth](#) via `-thread_depth`, or define new channels which one has added to the configuration to deal with additional load), it is desirable to make such changes "on the fly". This utility exists to allow making those changes that are both desirable, and feasible without too much disruption to existing Job Controller data structures.

58.1.5 Switches

58.1.5.1 `-channel=name`

Change a configuration setting for an existing [channel](#), or inform the Job Controller of the name of a new channel (of a valid, [already defined type](#)) that has been added to the configuration.

58.1.5.2 `-debug=n`

Set a [debug level](#) for Job Controller operation. `-global` must be specified in order to use `-debug`.

58.1.5.3 `-nodebug (default)`

Disable debugging for the Job Controller. This is default.

58.1.5.4 `-global`

Set certain global Job Controller options.

58.1.5.5 `-inorder_rebuild`

Rebuild message list in order. The `-global` switch must be specified in order to use `-inorder_rebuild`.

58.1.5.6 `-noinorder_rebuild (default)`

Do not rebuild the queues in order. This is the default.

58.1.5.7 `-job_limit=n`

The `-channel=name` switch must be specified in order to use `-job_limit`; override the effective [maxjobs/job_limit](#) for the specified channel.

58.1.5.8 `-max_messages=n`

Override the effective [max_cache_messages](#) value, or any previously set `-max_messages` value. `-global` must be specified in order to use `-max_messages`.

58.1.5.9 `-master_job=command`

Specify the command to execute (channel program to run) for the master direction of a channel. This switch may be used either with the `-channel` switch, to set values for a particular channel, or with the `-template` switch, to set values for a class of channels.

58.1.5.10 `-slave_job=command`

Specify the command to execute (channel program to run) for the slave direction of a channel. This switch may be used either with the `-channel` switch, to set values for a particular channel, or with the `-template` switch, to set values for a class of channels.

58.1.5.11 `-parallel_rebuild=n`

Override the value of the `rebuild_parallel_channels` Job Controller option. `-global` must be specified in order to use `-parallel_rebuild`.

58.1.5.12 `-template=channel-pattern`

Define a new type of channel, or redefine an existing type of channel. If redefining an existing type, all channels derived from that template are affected. The argument to `-template` should be a channel pattern; *e.g.*, `tcp_*`.

58.1.5.13 `-thread_depth=n`

The `-channel=name` switch must be specified in order to use `-thread_depth`; override the effective `threaddepth` for the specified channel.

58.1.6 Examples

```
# imsimta cache -change -global -debug=7
```

The above (UNIX) command turns on Job Controller debugging, at level 7.:

```
# imsimta cache -change -channel=tcp_special
```

This command informs the Job Controller of a new `tcp_special` channel that has been added to the configuration (in Unified Configuration, added using `msconfig` via an `edit channels` command or via appropriate `set channel:tcp_special.*` commands; in legacy configuration, added to the `imta.cnf` file). That is, this `imsimta cache -change` command does not in and of itself *define* the channel -- definition of the channel must be performed as normal; however, this command informs the Job Controller of the new channel, so that the Job Controller will know to begin running the channel, as needed.

58.2 cache -sync utility

Update the [Job Controller](#)'s in-memory cache of message files in the queues so as to reflect all messages currently present in the message queues.

58.2.1 Syntax

```
imsimta cache -synchronize
```

Table 58.2 imsimta cache -synchronize Command Switches

Switch	Default
-debug=n	-nodebug
-recipient	-recipient

58.2.2 Restrictions

Must have superuser privileges (UNIX) in order to use this utility.

58.2.3 Parameters

None.

58.2.4 Description

The `imsimta cache -synchronize` utility tells the MTA [Job Controller](#) to re-scan the queue disk area, `server-root/data/queue/*` on UNIX, checking for any (non-`.HELD`) message files not already present in the Job Controller's in-memory queue cache of message files.

58.2.5 Switches

58.2.5.1 -debug=n, -nodebug (default)

The `-debug` switch may be cause debugging of the cache synchronization. The default is `-nodebug`. Specifying `-debug` is equivalent to `-debug=1`.

58.2.5.2 -recipient (default), -norecipient

`-recipient`, which is the default, causes the cache synchronization to actually open message files to obtain full message details; this is slower than the "basic" operation (which can be selected with `-norecipient`), but gets more detailed message information.

58.2.6 Examples

To synchronize the queue cache, for instance after renaming a message file, issue the UNIX command

```
# imsimta cache -synchronize
```

58.3 cache -walk utility

Cause the Job Controller to write its current state (in particular, what message entries it has in its queue cache) to its log file.

58.3.1 Syntax

```
imsimta cache -walk
```

Table 58.3 imsimta cache -walk Command Switches

Switch	Default
-debug=n	-nodebug

58.3.2 Restrictions

Must have superuser privileges (UNIX) in order to use this utility.

58.3.3 Parameters

None.

58.3.4 Description

The `imsimta cache -walk` utility tells the [Job Controller](#) to write its current state (in particular, its current list of message entries in its queue cache) to its log file.

58.3.5 Switches

58.3.5.1 -debug=n, -nodebug (default)

The `-debug` switch may be used to cause the inclusion of additional debug output in state description that the Job Controller writes to its log file. The default is `-nodebug`.

58.3.6 Examples

This UNIX example shows telling the Job Controller to write its current state to its log file, with its debug level for this state dump set to 80:

```
# imsimta cache -walk -debug=80
```

58.4 chbuild utility

Compile the MTA character set conversion tables, and `tllds.txt` file.

58.4.1 Syntax

```
imsimta chbuild
```

Table 58.4 imsimta chbuild Command Switches

Switch	Default
<code>-image_file=file-spec</code>	<code>-image_file=IMTA_CHARSET_DATA</code>
<code>-maximum</code>	<code>-nomaximum</code>
<code>-option_file=file-spec</code>	<code>-nooption_file</code>
<code>-remove</code>	None
<code>-sizes</code>	<code>-nosizes</code>
<code>-statistics</code>	<code>-nostatistics</code>

58.4.2 Restrictions

Must have superuser privileges (UNIX) in order to use this utility.

58.4.3 Parameters

None.

58.4.4 Description

The `imsimta chbuild` utility compiles the character set conversion tables and loads the resulting image file into shared memory. As of MS 7.0.5, `imsimta chbuild` is also the means for informing the MTA of an updated `tllds.txt` file---the new file used as of MS 7.0.5 to store the IANA list of Top Level Domains.

The MTA ships with very complete character set tables so prior to MS 7.0.5, it was not normally necessary to run this utility. However, as of MS 7.0.5, `imsimta chbuild` should be used whenever an updated `tllds.txt` file has been fetched from IANA. (Note that to get updates to `tllds.txt` to take effect, it is not necessary to also do `imsimta cnbuild` after the `imsimta chbuild`---but it *is* necessary that processes be new in order to see the newly compiled charset/TLD data. So if in a hurry for the changes to take effect, issue an `imsimta restart` command.)

Prior to MS 7.0.5, two MTA Tailor options were relevant for `imsimta chbuild`: `imta_charset_data` specified the default output image file, and `imta_charset_option_file` specified an option file adjusting charset internal table sizes. As of MS 7.0.5, these MTA Tailor options have been deleted, and hard-coded file paths are used instead, `server-root/config/advanced/charset_data` and `server-root/lib/option_charset.dat`, where `server-root` is the value of the `SERVERROOT` environment variable.

58.4.5 Switches

58.4.5.1 `-image_file[=file-spec]`, `-noimage_file`

By default, `imsimta chbuild` creates as output the image file (formerly named by the `imta_charset_data` option of the MTA tailor file) `SERVERROOT/config/advanced/charset_data`. With the `-image_file` switch, an alternate file name may be specified. When the `-noimage_file` switch is specified, `imsimta chbuild` does not produce an output image file. This switch is used in conjunction with the `-option_file` switch to produce as output an option file which specifies table sizes adequate to hold the tables required by the processed input files.

58.4.5.2 `-maximum`, `-nomaximum` (default)

When `-maximum` is specified, the file `SERVERROOT/lib/maximum_charset.dat` is read, in addition to the charset option file (prior to MS 7.0.5 located via the `imta_charset_option_file` MTA Tailor option) `SERVERROOT/config/advanced/option_charset.dat`. This `maximum_charset.dat` file specifies near maximum table sizes but does not change any other charset option file (`option_charset.dat`) parameter settings. Only use the `-maximum` switch if the current table sizes are inadequate. The `-noimage_file` and `-option_file` switches should always be used in conjunction with this switch -- it makes no sense to actually output the enormous configuration that is produced by `-maximum`, but it does make sense to use `-maximum` to get past size restrictions in order to build an updated charset option file (`option_charset.dat`) containing proper size settings so that a properly sized character set image can be built with a subsequent `imsimta chbuild` invocation.

58.4.5.3 `-option_file[=file-spec]`, `-nooption_file` (default)

`imsimta chbuild` can optionally produce a charset option file that contains correct table sizes to hold the character set conversion tables which were just compiled (plus a little room for growth). The `-option_file` switch causes this file to be output, whereas use of the `-nooption_file` switch means that no option file will be output. Note that `imsimta chbuild` always *reads* any pre-existing charset option file (first looking for `SERVERROOT/config/advanced/option_charset.dat`, then `SERVERROOT/config/option_charset.dat`, then finally `SERVERROOT/lib/option_charset.dat` -- or prior to MS 7.0.5, looking for the file named by the `imta_charset_option_file` MTA Tailor option) -- use of the `-nooption_file` switch does not affect reading the old charset option file -- however, specifying `-option_file` causes `imsimta chbuild` to also *output* an updated charset option file. By default, the output charset option file is updated to the same location where the utility found the input charset option file. The value on the `-option_file` switch may be used to specify an alternate file name.

While the MTA ships with an initial `SERVERROOT/lib/option_charset.dat`, it is recommended that any site-generated `option_charset.dat` file instead be written to `SERVERROOT/config/advanced` (or at least to `SERVERROOT/config`) so that MTA updates will not overwrite the site-generated `option_charset.dat`. Thus the first time a site wishes to generate their own, modified `option_charset.dat`, the administrator should start by copying the distributed, initial `option_charset.dat` from the `SERVERROOT/lib` directory to the `SERVERROOT/config/advanced` directory.

The `-maximum` switch may be used in conjunction with `-option_file` to cause `imsimta chbuild` to read options from `maximum_charset.dat` in addition to the charset option

file (`option_charset.dat`). This `maximum_charset.dat` file specifies near maximum table sizes. Only use the `-maximum` switch if the current table sizes are quite inadequate, and only use it to create a new charset option file. The `-noimage_file` switch should always be specified when `-maximum` is specified since a maximum-size image would be truly enormous and extremely wasteful.

58.4.5.4 -remove

Remove any existant compiled character set conversion table; *i.e.*, remove the file (prior to MS 7.0.5 located via the `imta_charset_data` MTA Tailor option) `SERVERROOT/config/advanced/charset_data`.

58.4.5.5 -sizes, -nosizes (default)

The `-sizes` switch instructs `imsimta chbuild` to output information on the sizes of the uncompiled character set tables.

58.4.5.6 -statistics, -nostatistics (default)

The `-statistics` switch instructs `imsimta chbuild` to output information on the compiled conversion tables. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the `-option_file` switch is needed.

58.4.6 Examples

The standard command used on UNIX to compile character set conversion tables is:

```
# imsimta chbuild
```

Or if a drastically increased set of Top Level Domains (`tlds.txt` file) or drastically increased set of character sets calls for a big increase in the size of the compiled charset set tables, then:

```
# # If a site has never before generated their own option_charset.dat,
# #   first copy the distributed option_charset.dat to the site's config area
# cp SERVERROOT/lib/option_charset.dat SERVERROOT/config/advanced/option_charset.dat
# # Now have the chbuild utility figure out what it needs to resize...
# imsimta chbuild -noimage -option_file -maximum
# # Finally, build new compiled charset tables...
# imsimta chbuild
```

58.5 cnbuild utility

Legacy configuration: Compile the MTA configuration, alias, mapping, conversion, system wide filter, circuit check, and option files, and various `configutil` parameters, as well as (optionally) the reverse "database", general "database", and forward "database", and also (as of MS 7.0) the Job Controller configuration file, the Dispatcher configuration file, and TCP/IP channel option files; generate an image file (suitable for memory mapping by MTA processes).

Unified Configuration: Compile `confdesc.xml` as well as (optionally) the reverse "database", general "database", and forward "database"; generate an image file (suitable for memory mapping by MTA processes).

58.5.1 Syntax

```
imsimta cnbuild
```

Table 58.5 imsimta cnbuild Command Switches

Switch	Default
<code>-image_file=file-spec</code>	<code>-image_file=IMTA_ROOT:config/advanced/config_data</code>
<code>-maximum</code>	<code>-nomaximum</code>
<code>-option_file=file-spec</code>	<code>-option_file=IMTA_ROOT:/config/option.dat</code>
<code>-remove</code>	None
<code>-sizes</code>	<code>-nosizes</code>
<code>-statistics</code>	<code>-nostatistics</code>
<code>-check</code>	<code>-nocheck</code>
<code>-synonyms</code>	<code>-nosynonyms</code>
<code>-xml_config</code>	<code>-noxml_config</code>

58.5.2 Restrictions

Must have superuser privileges (UNIX) in order to use this utility.

58.5.3 Parameters

None.

58.5.4 Description

With a legacy configuration, the `cnbuild` utility compiles the textual configuration, option, mapping, conversion, system wide filter, circuit check, and alias files, and various `configutil` parameters (see [Basic configuration settings relevant to alias LDAP lookups](#) and [Basic configuration settings relevant to domain LDAP lookups](#)), and (depending upon the setting of the `use_text_databases` MTA option) also optionally the [reverse "database"](#), [general "database"](#), and [forward "database"](#), and (as of MS 7.0) also the [Job Controller configuration file](#), [Dispatcher configuration file](#), and [TCP/IP channel option files](#) (for SMTP

channels or LMTP client channels), and generates an image file (suitable for memory mapping by MTA processes). With a Unified Configuration, most of these files have been consolidated: when using a Unified Configuration, the `cnbuild` utility compiles the `confdesc.xml` file, and (depending upon the setting of the `use_text_databases` MTA option) also optionally the `reverse "database"`, `general "database"`, and `forward "database"`, and generates an image file (suitable for memory mapping by MTA processes). The resulting image is the file formerly named by the `imta_config_data` option of the MTA tailor file; as of MS 7.0.5, such MTA tailor file options have been replaced by consistent locations, and the image file is simply `IMTA_ROOT:config/advanced/config_data`.

Whenever a component of the MTA (*e.g.*, a channel program) must read any possibly compiled configuration component, it first checks to see whether a compiled configuration image file exists, and if so, the running program uses that image. There are five exceptions to this rule. The first is `imsimta cnbuild` itself, which for obvious reasons always reads the text files and never tries to use an image form of the configuration data. The remaining four exceptions are `imsimta test -domain_map`, `imsimta test -rewrite`, `imsimta test -mapping`, and `imsimta test -x400` which can all be instructed with the `-image_file` switch to use a different compiled configuration file. This facility in `imsimta test -rewrite` is useful for testing changes prior to compiling them.

One reason for compiling configuration information is simple: performance. Precompiling the MTA configuration allows MTA processes to quickly load that precompiled configuration and then get immediately to work, rather than each new MTA process needing to read the MTA configuration and build the in-memory representation of the configuration itself, before starting its own work. Note that MTA design includes short-lived, transient processes (*e.g.*, most channel delivery job processes), as well as long-lived processes (*e.g.*, [Dispatcher](#) and [Job Controller](#)), so avoiding the overhead of each MTA process, short-lived or not, repeating the same configuration-processing work reduces overhead.

A second, and nowadays often more important, reason for compiling configuration information is for convenience in configuration management and updating: due to the fact that when a compiled configuration exists the MTA processes will normally refer to it preferentially, the MTA administrator can make changes to the underlying configuration files at leisure, and optionally perform some testing of such changes (*e.g.*, using a `-noimage_file` switch with test utilities) prior to compiling the updated configuration and having the changes become "live"; meantime, the MTA continues running with the older, compiled configuration.

Once you begin to use a compiled configuration, it will be necessary to recompile the configuration every time changes are made to the settings and files comprising the source of the compiled configuration. For a Unified Configuration, this is primarily `confdesc.xml` (normally modified using the `msconfig` utility); if the `use_text_databases` MTA option has been set, then also potentially the `reverse "database"` replacement text file, the `general "database"` replacement text file, and the `forward "database"` replacement text file. Specifically, these are the files

- `SERVERROOT/config/confdesc.xml`,
- `SERVERROOT/config/reverse.txt`,
- `SERVERROOT/config/general.txt`, and
- `SERVERROOT/config/forward.txt`.

For a legacy configuration, this includes various relevant `configutil` parameters (see [Basic configuration settings relevant to alias LDAP lookups](#) and [Basic configuration settings](#)

relevant to domain LDAP lookups) and the following files: the MTA configuration file (or any files referenced by it, such as `internet.rules`), the MTA system alias file, the MTA mapping file, the MTA option file, the MTA conversion file, system wide filter file, or the circuit check configuration file; if the `use_text_databases` MTA option has been set, then also the reverse "database" replacement text file, the general "database" replacement text file, the forward "database" replacement text file; as of MS 7.0, also the Job Controller configuration file, the Dispatcher configuration file, and TCP/IP channel option files (affecting SMTP channels and LMTP client channels). Specifically, these are the files which, prior to MS 7.0.5, were pointed at by the MTA Tailor file options `imta_config_file`, `imta_alias_file`, `imta_mapping_file`, `imta_option_file`, `imta_conversion_file`, `imta_system_filter_file`, `imta_reverse_data`, `imta_general_data`, `imta_forward_data`, respectively, or nowadays are simply:

- `SERVERROOT/config/imta.cnf`,
- `SERVERROOT/config/aliases`,
- `SERVERROOT/config/mappings`,
- `SERVERROOT/config/option.dat`,
- `SERVERROOT/config/conversions`,
- `SERVERROOT/config/imta.filter`,
- `SERVERROOT/config/reverse.txt`,
- `SERVERROOT/config/general.txt`, and
- `SERVERROOT/config/forward.txt`,

as well as the file `SERVERROOT/config/circuitcheck.cnf`.

Until such time as the configuration is recompiled, changes to any of these files will not be visible to the MTA. Furthermore, even after recompiling the configuration, note that changes to the MTA compiled configuration will not be seen by *running* MTA processes unless and until the MTA configuration is reloaded via the `imsimta reload` utility, or until the process expires and is replaced by a new process. In the case of transient, short-lived processes, (*e.g.*, most channel delivery jobs) their own natural quick expiration may lead to them being replaced by new processes soon enough that any lag in making use of the new MTA configuration is relatively unimportant; however, for significant changes impacting the **Job Controller**, or the **Dispatcher** or its server processes (SMTP server processes, SMTP SUBMIT server processes, LMTP server processes), after performing `imsimta cnbuild` also consider performing `imsimta reload`, when the change needs to take effect "immediately" and `imsimta reload` will suffice. (Note that `imsimta reload` only updates certain parts of the precompiled MTA configuration; in particular, it does not reload the channel definitions and rewrite rules portions of the MTA configuration as they are complex and intertwined and require a full re-read of the configuration. Some additional cases of configuration changes relevant to the Job Controller can instead be modified "live", without requiring a restart of the Job Controller, via the `imsimta cache -change` utility.) A full restart of the MTA (especially restarting the Job Controller) should be avoided on production systems unless truly necessary as a restart interrupts and disrupts message delivery; however, a `imsimta restart *` to restart the various servers underneath the Dispatcher causes only brief disruption in acceptance of incoming messages, so is acceptable when enqueue processing

changes beyond those covered by `imsimta reload` need to take effect "immediately" upon incoming messages.

Note that updates to the `tlds.txt` file (introduced in MS 7.0.5) do *not* require use of `imsimta cnbuild` to take effect; instead, updates to `tlds.txt` are incorporated by use of the `chbuild` utility.

See [Compiling the MTA configuration](#) for further details on the use of compiled configurations.

If `imsimta cnbuild` spots a syntactic error in one of the files that it is attempting to use to build the compiled configuration, it will be reported, typically as an error of the form

```
time-stamp: Error in mm_init -- detail
```

where the *detail* provides additional, specific detail about the particular error. For instance (with output wrapped here for display convenience; in reality it would appear all on one line):

```
09:28:26.15: Error in mm_init -- duplicate host in channel table -- host.domain  
com -- line #45 in file IMTA_CONFIG_FILE
```

See the discussion of such errors in Section 29.2.2.3.1 for examples and discussion of details of configuration syntax errors. Note that `imsimta cnbuild` does not know specifics of the syntax and semantics of the Dispatcher configuration file or Job Controller configuration file, or channel option files, so generally errors in the option settings in those files---other than egregiously, obviously broken syntax problems---will not be reported by the utility and instead will get reported by the component in question when it attempts to begin running.

As of JES MS 6.3p1, errors compiling the configuration (in particular, `mm_init` errors) will cause `imsimta cnbuild` to exit with a non-zero status. Prior to that version, while such errors would certainly cause `imsimta cnbuild` to exit with error text and without making a new compiled configuration, the exit status was nevertheless zero in many cases.

There is also a secondary, "tuning" use of `imsimta cnbuild`. When the MTA is building an in-memory representation of its configuration (whether that is to generate a compiled configuration, or whether in the absence of any compiled configuration MTA processes are each reading the MTA configuration and building their own private representations of the MTA configuration), it starts by assuming certain table sizes, and if those table sizes are not adequate, iteratively increases those sizes until reaching "big enough" tables in memory to hold the current configuration. The starting points for such table sizes are controlled by [internal size MTA options](#). Using `imsimta cnbuild` with its `-statistics` switch provides information on how "close" the current internal size MTA options are to the sizes needed for the current MTA configuration, and using `imsimta cnbuild` with the set of switches `-noimage_file -maximum -option_file` will generate a new MTA option file with internal size MTA options set appropriately for the current MTA configuration. This sort of tuning permits more efficient compilation of the MTA configuration---though it is not strictly necessary (since the MTA will iteratively resize, if necessary, on-the-fly while reading its configuration).

Since the MTA will resize its internal table sizes as needed, errors about exceeding table sizes are normally seen only if the MTA's more-or-less "hard" limits on resizing are reached. (The limits are established by the `maximum.dat` file and/or "hard" limits in the code.) And since the MTA's "hard" limits are *very* generous, exceeding the limits is usually an indication of either a configuration error or a type that has confused the MTA about the intended meaning

of certain configuration inputs (for instance, an extraneous blank line in the rewrite rules, causing the MTA to attempt to interpret all remaining material as channel definitions), or configuration choices involving poor use of MTA facilities that would be better handled in an alternate manner (such as attempting to [hard code many thousands of mapping table entries](#), rather than using a few general entries that do [general database callouts](#) for the specific fields). In particular, reaching the limits specified in the normal `maximum.dat` file is usually an indication of poor configuration choices; you should contact Oracle if you believe you wish to exceed those limits, as you may be better served by alternate configuration tactics. See also Section 30.3.1 for a discussion of `mm_init` errors, and some of the typical configuration errors that can cause them.

Finally, a command such as

```
# imsimta cnbuild -noimage_file -option_file=file-spec
```

may be used to list current option values (as specified in the current MTA option file, plus default values) in the file specified as `file-spec`. To list (most) default option values, irrespective of override values specified in the current MTA option file, the current MTA option file must be "moved aside" before the above command is issued. (For instance, temporarily rename the MTA option file, then issue the above command, then rename the MTA option file back to its normal location.)

58.5.5 Switches

58.5.5.1 -check, -nocheck (default)

`-check` means to check the structure of the `confdesc.xml` file.

58.5.5.2 -image_file[=file-spec], -noimage_file

By default, `imsimta cnbuild` creates as output the image file named `SERVERROOT/config/advanced/config_data`; (prior to MS 7.0.5, the file located via the [imta_config_data](#) MTA Tailor option). With the `-image_file` switch, an alternate file name may be specified. When the `-noimage_file` switch is specified, `imsimta cnbuild` does not produce an output image file. This switch is used in conjunction with the `-option_file` switch to produce as output an option file which specifies table sizes adequate to hold the configuration required by the processed input files.

58.5.5.3 -maximum, -nomaximum (default)

When `-maximum` is specified, the file `SERVERROOT/lib/maximum.dat` is read in addition to the [internal size MTA options](#) (which in legacy configuration would be stored in the `MTA option.dat` file, located prior to MS 7.0.5 via the `imta_option_file` MTA Tailor file option). The `maximum.dat` file specifies near maximum table sizes but does not change any other MTA option settings. Only use the `-maximum` switch if the current table sizes are inadequate. The `-noimage_file` and `-option_file` switches should always be used in conjunction with this switch---it makes no sense to output the enormous configuration that is produced by `-maximum`, but it does make sense to use `-maximum` to get past size restrictions in order to set properly adjusted MTA option values (in legacy configuration, build an MTA option file with properly adjusted option values) so that a properly sized configuration can be built with a subsequent `imsimta cnbuild` invocation.

58.5.5.4 `-option_file[=file-spec]`, `-nooption_file` (default)

`imsimta cnbuild` can optionally set various internal size MTA options (in legacy configuration, produce an option file that contains correct table sizes) to hold the configuration that was just compiled (plus a little room for growth). The `-option_file` switch causes this file to be output. By default, this file is named `SERVERROOT/config/option.dat` (or prior to MS 7.0.5, was located via the `imta_option_file` MTA Tailor file option). The value on the `-option_file` switch may be used to specify an alternate file name. If the `-nooption_file` switch is given, then no option file will be output. `imsimta cnbuild` always reads any MTA options previously set, whether set as Unified Configuration [MTA options](#) or set in the MTA option file, `SERVERROOT/config/option.dat`; use of the `-nooption_file` switch will not alter this behavior. However, use of the `-maximum` switch causes `imsimta cnbuild` to also read MTA options from the file `SERVERROOT/config/advanced/maximum.dat`. This `maximum.dat` file specifies near maximum table sizes. Only use the `-maximum` switch if the current table sizes are inadequate, and only use it to generate new MTA option settings (set either as new values for [MTA options](#) in Unified Configuration, or by building a new `option.dat` file in legacy configuration). The `-noimage_file` switch should always be specified when `-maximum` is specified since a maximum-size image would be truly enormous and extremely wasteful.

58.5.5.5 `-remove`

Remove any existant compiled configuration; *i.e.*, remove the file `SERVERROOT/config/advanced/config_data` (located prior to MS 7.0.5 via the `imta_config_data` MTA Tailor file option).

58.5.5.6 `-sizes`, `-nosizes` (default)

The `-sizes` switch instructs `imsimta cnbuild` to output information on the sizes of uncompiled MTA tables.

58.5.5.7 `-statistics`, `-nostatistics` (default)

The `-statistics` switch instructs `imsimta cnbuild` to output information on how much of the various tables in the compiled configuration were actually used to store data. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the `-option_file` switch is needed. Specifying `-statistics` effectively forces `-noimage_file`.

58.5.5.8 `-synonyms`, `-nosynonyms` (default)

`-synonyms` means to output a list of MTA option and channel option synonyms; that is, output a list of the MTA options and channel option marked as aliases in `confdesc.xml`:

```
spamfilter_config_file -> spamfilter1_config_file
brightmail1_config_file -> spamfilter1_config_file
brightmail_config_file -> spamfilter1_config_file
...additional aliased MTA options...
include_spares -> include_spares1
733 -> percents
822 -> sourceroute
brightmail -> sourcespamfilter1
```

```
channelfilter -> destinationfilter
...additional aliased channel keywords...
```

58.5.5.9 -xml_config, -noxml_config (default)

-xml_config directs cnbuild to compile a Unified Configuration; -noxml_config directs cnbuild to compile a legacy configuration.

58.5.6 Examples

#1

```
# imsimta cnbuild
# # Depending upon what has changed, consider issuing:
# # imsimta reload
# # imsimta restart *
```

This is the standard command used on UNIX to regenerate a compiled configuration. After compiling the configuration, restart any programs which may need to reload the new configuration; *e.g.*, the SMTP server should be restarted. (Note that the * character may need shell-quoting so that it will be passed through the shell properly to the MTA utility.)

#2

```
# imsimta cnbuild -noimage_file -option_file -maximum
# imsimta cnbuild
```

Use these two UNIX commands when you encounter the infamous "No room in table" error message.

58.6 crdb utility

`imsimta crdb` is a utility used to create and update MTA (on-disk) database files.

58.6.1 Syntax

```
imsimta crdb input-file-spec output-database-spec
```

Table 58.6 imsimta crdb Command Switches

Switch	Default
<code>-append</code>	<code>-noappend</code>
<code>-count</code>	<code>-count</code>
<code>-dump</code>	See text
<code>-duplicates</code>	<code>-noduplicates</code>
<code>-exception_file=<i>file-spec</i></code>	<code>-noexception_file</code>
<code>-huge_records</code>	<code>-huge_records</code>
<code>-long_records</code>	<code>-nolong_records</code>
<code>-quoted</code>	<code>-noquoted</code>
<code>-remove</code>	<code>-noremove</code>
<code>-statistics</code>	<code>-statistics</code>
<code>-strip_colons</code>	<code>-nostrip_colons</code>

58.6.2 Restrictions

None.

58.6.3 Parameters

58.6.3.1 input-file-spec

A text file containing the entries to be placed into the database. Each line of the text file must correspond to a single entry.

58.6.3.2 output-database-spec

The initial name string of the file to which to write the database; the database will consist of a file named `output-database-spec.db`.

58.6.4 Description

`imsimta crdb` is a utility to create and or update MTA (on disk) database files. `imsimta crdb` simply converts a plain text file into MTA database records and from them either creates a new database or adds the records to an existing database.

On UNIX, if run from the `root` account, `imsimta crdb` will set the ownership of the database it creates to the MTA user account; see the `user` option from the `restricted.cnf` file, or in older versions of the MTA, see the `imta_user` MTA Tailor option. If run from an unprivileged account, then the database will be owned by that unprivileged user.

In general, each line of the input file must consist of a left hand side and a right hand side. The two sides are separated by one or more spaces or tabs. The left hand side is limited to 32 characters in a short database (the default variety) and 80 characters in a long database. The right hand side is limited to 80 characters in a short database and 256 in a long database. Spaces and tabs may not appear in the left hand side (but see the description of the `-quoted` switch below).

The format of the input file is described in the sections describing each particular MTA database. For instance, the format of the input file for an alias database is described in [Alias database format](#); the format of the input file for the domain database (rewrite rule database) is described in [Domain database](#); the format of the input file for the forward database is described in [Forward database](#); the format of the input file for the general database is described in [General database](#); the format of the input file for the address reversal database is described in [Reverse database](#).

58.6.5 Switches

58.6.5.1 `-append`, `-noappend` (default)

When the default, `-noappend`, switch is in effect, a new database is created, overwriting any old database of that name. Use the `-append` switch to instruct the MTA to instead add the new records to an existing database.

58.6.5.2 `-count` (default), `-nocount`

Controls whether or not a count is output after each group of 100 input lines are processed.

58.6.5.3 `-dump`

`imsimta crdb -dump` is a synonym for `imsimta dumpdb`. It is used to dump an existing database to a flat text file---or to stdout if no output file is specified. The parameters are interpreted as the input database specification, and optionally a flat text file to which to write the output. No other switches are valid when `-dump` is specified.

58.6.5.4 `-duplicates`, `-noduplicates` (default)

Controls whether or not duplicate records are allowed in the output files. Currently duplicate records are of use only in the domain database (rewrite rules database) and databases associated with the directory channel.

58.6.5.5 `-exception_file=file-spec`, `-noexception_file` (default)

`imsimta crdb` may encounter records that cannot be loaded into the database. This usually means that these records had keys (left hand sides) that were duplicates of other keys previously encountered in the input file. These exception records can optionally be written to a separate output file for later examination; the `-exceptions_file` switch controls the writing

of this file. Note that the lines in this file are not plain text; they are formatted as database entries.

58.6.5.6 `-long_records`, `-nolong_records` (default), `-huge_records`, `-nohuge_records`

These switches control the size of the output records. By default left hand sides are limited to 32 characters and right hand sides are limited to 80 characters. If `-long_records` is specified, the limits are changed to 80 and 256, respectively. If `-huge_records` is specified, the limits are changed to 252 and 1024, respectively. Currently, `-huge_records` databases are supported only for the alias database.

58.6.5.7 `-quoted`, `-noquoted` (default)

This switch controls the handling of quotes. Normally `imsimta crdb` pays no particular attention to double quotes. If `-quoted` is specified, `imsimta crdb` matches up double quotes in the process of determining the break between the left and right hand sides of each input line. Spaces and tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. Note: The quotes are not removed unless the `-remove` switch is also specified.

58.6.5.8 `-remove`, `-noremove` (default)

These switches control the removal of quotes. If `imsimta crdb` is instructed to pay attention to quotes, the quotes are normally retained. If `-remove` is specified, `imsimta crdb` removes the outermost set of quotes from the left hand side of each input line. Spaces and tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. Note: `-remove` is ignored if `-quoted` is not in effect.

58.6.5.9 `-statistics` (default), `-nostatistics`

Controls whether or not some simple statistics are output by `imsimta crdb`, including the number of entries (lines) converted, the number of exceptions (usually duplicate records) detected, and the number of entries that could not be converted because they were too long to fit in the output database. `-nostatistics` suppresses output of this information.

58.6.5.10 `-strip_colons`, `-nostrip_colons` (default)

The `-strip_colons` switch instructs `imsimta crdb` to strip a trailing colon from the right end of the left hand side of each line it reads from the input file. This is useful for turning [alias file](#) entries into an [alias database](#).

58.6.6 Examples

```
# imsimta crdb -long_records IMTA_ROOT:data/db/aliases.txt IMTA_ROOT:data/db/tmpdb
# imsimta renamedb IMTA_ROOT:data/db/tmpdb IMTA_ROOT:data/db/aliasesdb
```

The above example shows UNIX commands that may be used to create an [alias database](#) with "long" record entries; note that the creation is performed in a two-step process using a temporary database to minimize any window of time, such as during database generation, when the database would be locked and inaccessible to the MTA.

58.7 process utility

List currently executing MTA processes and jobs.

58.7.1 Syntax

```
imsimta process
```

58.7.2 Restrictions

None.

58.7.3 Parameters

None.

58.7.4 Description

Show current MTA processes. Normally on a regular MTA system, the Dispatcher and Job Controller should always be present; typically some SMTP and SMTP SUBMIT server processes are present; and additional processes may be present if messages are currently being processed, or if certain additional MTA components are in use. On an LMTP back end Message Store system, the Dispatcher should always be present, and typically one or more LMTP server processes are present.

58.7.5 Examples

The following command shows currently executing Messaging Server MTA processes on a regular MTA system:

```
# imsimta process
  USER  PID S  VSZ  RSS   STIME      TIME COMMAND
mailsrv  235 S  44592 20324 18:25:00    00:01 /opt/sun/comms/messaging64/lib/tcp_smtp_server
mailsrv  236 S  44756 20924 18:25:00    00:04 /opt/sun/comms/messaging64/lib/tcp_smtp_server
mailsrv  262 S  44816 21024 18:27:30    00:01 /opt/sun/comms/messaging64/lib/tcp_smtp_server
mailsrv  358 S  44836 21052 18:37:30    00:04 /opt/sun/comms/messaging64/lib/tcp_smtp_server
mailsrv  8286 S  44296 19724 08:55:08    00:00 /opt/sun/comms/messaging64/lib/managesieve
mailsrv  8287 S  44232 19648 08:55:08    00:00 /opt/sun/comms/messaging64/lib/managesieve
mailsrv 18763 S  42068 9004   Aug_23    03:44 /opt/sun/comms/messaging64/lib/dispatcher
mailsrv 18775 S  37492 12044   Aug_23    00:06 /opt/sun/comms/messaging64/lib/job_controller
```

58.8 purge utility

Purge MTA log files.

58.8.1 Syntax

```
imsimta purge [file-pattern]
```

Table 58.7 imsimta purge Command Switches

Switch	Default
-day= <i>value</i>	None
-debug	None
-hour= <i>value</i>	None
-num= <i>value</i>	-num=5

58.8.2 Restrictions

On UNIX, must have write access to the directory containing the file(s) to be purged.

58.8.3 Parameters

58.8.3.1 file-pattern

A file name pattern for which MTA log files to purge. The default, if no file name pattern is specified, is to purge *all* the files in the MTA log directory. If no directory path is included in the file name pattern, the default is to purge files matching the file name pattern from the MTA log directory.

58.8.4 Description

`imsimta purge` purges back older versions of MTA log files. (`imsimta purge` can tell which log files are older based on the `uniqueid` strings terminating MTA log file names.)

58.8.5 Switches

58.8.5.1 -day=*d*

Specifying `-day=d` results in purging all but the last *d* days worth of log files. Note that here "day" means a 24 hour period, rather than a calendar day (midnight to midnight); *i.e.*, all but the log files created in the last 24*d* hours will be purged.

58.8.5.2 -debug

On UNIX, specifying `-debug` enables some debug output to stdout.

58.8.5.3 -hour=*h*

Specifying `-hour=h` results in purging all but the last *h* hours worth of log files.

58.8.5.4 `-num=n`

Specifying `-num=n` results in purging all but the last `n` log files. The default is `-num=5`.

58.8.6 Examples

```
# imsimta purge
```

This UNIX command will purge all but the last five versions of each sort of log file in the MTA log directory, `SERVERROOT/log`.

```
# imsimta purge -num=10 tcp_local_master.log
```

This UNIX command will purge all but the last ten versions of any `tcp_local_master.log-*` files from the MTA log directory, `SERVERROOT/log/`.

58.9 reload utility

Reload portions of the MTA configuration.

58.9.1 Syntax

```
imsimta reload
```

58.9.2 Restrictions

On UNIX, must have superuser privileges in order to use this utility.

58.9.3 Parameters

None.

58.9.4 Description

Currently, this utility reloads (updates the in-memory copy of the compiled configuration) the mapping file, and the general, reverse, and forward "database", if the applicable bits of [use_text_databases](#) have been set. It does not, however, reload the entire compiled configuration; in particular, changes to the `imta.cnf` file are not propagated by `imsimta reload`.

For making certain changes "live" to a running [Job Controller](#), see also the [cache -change](#) utility.

58.10 test -match utility

Test a [mapping wildcard pattern](#).

58.10.1 Syntax

```
imsimta test -match
```

58.10.2 Restrictions

None.

58.10.3 Parameters

None.

58.10.4 Description

`imsimta test -match` may be used to test a mapping pattern, particularly, to test wildcard and glob matching.

When invoked, `imsimta test -match` prompts for a pattern and then for a target string to compare against the pattern, and will output whether or not the target string matched and if it did match, which characters in the target string matched which wildcard or glob of the pattern. `imsimta test -match` will loop, prompting for input, until exited with a CTRL/D (UNIX).

58.10.5 Examples

```
% imsimta test -match
Pattern: [$[ax1]*@*.acme.com
[ 1S] cglob [1ax]
[ 2] "@"
[ 3S] glob, req 109, reps 2
[ 4] "."
[ 5] "a"
[ 6] "c"
[ 7] "m"
[ 8] "e"
[ 9] "."
[ 10] "c"
[ 11] "o"
[ 12] "m"
Target: xx11a@sys1.acme.com
Match.
0 - xx11a
1 - sys1
Pattern: [$[ax1]*@*.acme.com
[ 1S] cglob [1ax]
```

```

[ 2] "@"
[ 3S] glob, req 109, reps 2
[ 4] "."
[ 5] "a"
[ 6] "c"
[ 7] "m"
[ 8] "e"
[ 9] "."
[ 10] "c"
[ 11] "o"
[ 12] "m"
Target: 12a@node.acme.com
No match.
Pattern: $(ax1)*@*.acme.com
[ 1S] cglob [1ax]
[ 2] "@"
[ 3S] glob, req 109, reps 2
[ 4] "."
[ 5] "a"
[ 6] "c"
[ 7] "m"
[ 8] "e"
[ 9] "."
[ 10] "c"
[ 11] "o"
[ 12] "m"
Target: 1xa@node.acme.com
Match.
0 - 1xa
1 - node
Pattern: ^D
%
```

In the above UNIX example, the sample mapping pattern `$(ax1)*@*.acme.com` is tested for several sample target strings.

```

% imsimta test -match
Pattern: $(1.2.3.0/24)
[ 1S] ipv4 [1.2.3.0/255.255.255.0]
Target: 1.2.3.4
Match.
0 - 1.2.3.4
Pattern: $(1.2.3.0/24)
[ 1S] ipv4 [1.2.3.0/255.255.255.0]
Target: 1.2.8.0
No match.
Pattern: ^D
%
```

In the above UNIX example, the sample mapping pattern `$(1.2.3.0/24)` is tested for two sample target strings.

58.11 test -rewrite utility

Test address rewriting specified by an MTA configuration.

58.11.1 Syntax

```
imsimta test -rewrite [test-address]
```

Table 58.8 imsimta test -rewrite Command Switches

Switch	Default
-aby= <i>value</i>	-noaby
-additions	-noadditions
-alias_file= <i>file-spec</i>	-alias_file=IMTA_ROOT:config/aliases
-alternate_recipient= <i>address</i>	-noalternate_recipient
-by= <i>value</i>	-noby
-applicationinfo= <i>string</i>	None
-channel	-channel
-check_expansions	-nocheck_expansions
-configuration_file= <i>file-spec</i>	-configuration_file=IMTA_ROOT:config/imsimta.cnf
-conversion_file= <i>file-spec</i>	-conversion_file=IMTA_ROOT:config/conversions
-database= <i>database-list</i>	See text
-debug	-nodebug
-delivery_receipt	See text
-destination_channel= <i>channel</i>	None
-ebm	-noebm
-esmtpush	-noesmtpush
-expandlimit= <i>n</i>	None
-extra_local_channel= <i>channel-name</i>	None
-filter	-nofilter
-from= <i>address</i>	-from=postmaster@localhost
-header	-noheader
-identifiers	-noidentifiers
-image_file[= <i>file-spec</i>]	-image_file

-input= <i>input-file-spec</i>	-input=stdin
-jacket	-nojacket
-lmtptused	-nolmtptused
-local_alias= <i>value</i>	-nolocal_alias
-mapping_file= <i>file-spec</i>	-mapping_file=IMTA_ROOT:config/mappings
-mtpriority= <i>n</i>	-mtpriority=0
-option_file= <i>file-spec</i>	-option_file=IMTA_ROOT:config/option.dat
-output= <i>output-file-spec</i>	output=stdout
-password= <i>string</i>	-nopassword
-proxyused	-noproxyused
-read_receipt	See text
-reprocessing	-reprocessing
-restricted= <i>setting</i>	-restricted=0
-rrvs= <i>ISO8601-value</i>	-norrvs
-saslused	-nosaslused
-sender= <i>address</i>	-nosender
-size= <i>n</i>	-nosize
-soptin	-nosoptin
-source_channel= <i>channel</i>	-source_channel=1
-spares	-nospares
-statistics	-nostatistics
-system_filter= <i>file-spec</i>	-system_filter=IMTA_ROOT:config/imta.filter
-tag= <i>tag-list</i>	-notag
-tlsused	-notlsused
-transportinfo= <i>string</i>	None
-user= <i>string</i>	-user="--USERNAME--"
-utf8	-noutf8
-xml_config= <i>file-spec</i>	-xml_config=IMTA_TABLE:config.xml

58.11.1.1 Restrictions

Must be superuser or the MTA user (the user option in `restricted.cnf`, or prior to MS 7.0.5, the `imta_user` MTA Tailor option value), or be in the group specified by the `group` option in `restricted.cnf` (prior to MS 7.0.5, be in the `imta_world_group` group), in order to display `-filter`, `-soptin`, or `-spares` output.

58.11.1.2 Prompts

Address:	<i>test-address</i>
----------	---------------------

58.11.2 Parameters

58.11.2.1 *test-address*

Optional parameter specifying one or more (comma-separated) addresses to rewrite.

58.11.3 Description

`imsimta test -rewrite` provides a straightforward test facility for examining the MTA's address rewriting and channel matching process without actually sending any message. Various qualifiers can be used to control whether `imsimta test -rewrite` uses the configuration text files or the compiled configuration (if present), the amount of output produced, and so on.

The `imsimta test -rewrite` utility has several especially common and useful uses:

1. testing overall *syntactic* validity (though not *semantic* correctness) of the MTA configuration,
2. testing MTA configuration changes prior to making them "live" with `imsimta cnbuild`, and conversely checking that the compiled configuration in fact corresponds to the "most current" versions of the configuration files,
3. testing that the Messaging Server LDAP configuration information is accessible,
4. testing that the LDAP server is responding to domain and user/group lookups,
5. testing the rewriting, [alias lookup and expansion](#), and resulting routing, of specific addresses,
6. testing the expansion (membership) of [groups and mailing lists](#),
7. testing the effects of address-based `*_ACCESS` mapping tables,
8. testing the effects of posting restrictions such as restrictions on mailing list postings,
9. determining which [Sieve filters](#) are applicable for a particular recipient address.

If a test address is specified on the command line, `imsimta test -rewrite` applies MTA address rewriting to that address, reports the results, and exits. If no test address is specified, `imsimta test -rewrite` will enter a loop, prompting for an address, rewriting it, and prompting again for another address. `imsimta test -rewrite` will exit when CTRL/D (UNIX) is entered.

In interactive mode, note that the caret character, `^`, may be used to enter a character or characters by ASCII value (in hexadecimal); each character must be entered as a two digit hexadecimal value, with a final caret character meaning to return to "normal" (as typed) character entry. For instance, `^20^` is one way of entering a space character. To enter a literal caret character, caret-quote the caret, `^^`.

When testing an alias corresponding to a mailing list which has an [AUTH_ or CANT_ type of named parameter](#) (legacy configuration) or an `alias_auth_*` or `alias_cant_*` alias option (Unified Configuration) controlling who may post to the list, or which has an `mgrp[Dis]Allowed*` LDAP attribute controlling who may post to the list, or when testing rewriting when [SEND_ACCESS or related mapping tables](#) are in effect, note that by default

`imsimta test -rewrite` uses as the posting address the return address of the [local postmaster](#) as specified by the [return_address](#) MTA option. To specify a different posting address for the rewriting process, use the `-from` switch.

Note that as mentioned above, the `imsimta test -rewrite` utility also provides a basic "sanity check" of the syntactical correctness (though not the semantic correctness) of the configuration. In particular, if the utility returns any

```
Error in mm_init -- detail
```

error message, that is a warning of a serious configuration problem, preventing the MTA from operating.

If an active compiled configuration appears to be "out-of-date" compared to configuration files, then the `imsimta test -rewrite` utility will issue the following warning (but proceed to operate):

```
Warning - compiled configuration does not match configuration files
-- detail
```

with further detail (such as what file(s) appear to have been modified subsequent to the compilation of the currently active compiled configuration) in the `detail` text.

The `imsimta test -rewrite` utility also provides a way of checking that the LDAP server is responsive. Using the `-noimage` switch, that is, using an `imsimta test -rewrite -noimage` command, is a way of checking that the Messaging Server configuration information in LDAP is accessible; if it is not, then the utility will return a warning of the general form:

```
[date-and-time] hostname [pid]: General Warning: could not get server configuration in ldap, using cached configuration information
```

and then proceed to attempt to process the address using cached LDAP configuration information. Note that one of the more common cases where the above warning can be issued is where the LDAP server is in fact not responding, in which case the `imsimta test -rewrite` utility may further not be able to successfully lookup "local" domains and users. Then temporary errors of the form

```
4.0.0 Temporary lookup failure: address
```

(or whatever is configured via the [domain_failure](#) MTA option) when attempting to rewrite addresses suggest that the LDAP user/group directory is unavailable/unresponsive; further details on the underlying LDAP error may be obtained using the utility's `-debug=level=3` switch.

58.11.4 Switches

58.11.4.1 `-additions`, `-noadditions` (default)

(New in MS 8.0) Specifying `-additions` causes any added prefix or suffix text to be displayed. `-noadditions` is the default.

58.11.4.2 **-alias_file=filename**

If a compiled configuration is not being used, then `imsimta test -rewrite` normally consults the default alias file during the rewriting process. Prior to MS 7.0, that alias file was located via the `imta_alias_file` option of the MTA Tailor file, so usually `/opt/SUNWmsgsr/config/aliases`; as of MS 7.0, the alias file is located as `SERVERROOT/config/aliases`. The `-alias_file` qualifier specifies an alternate file for `imsimta test -rewrite` to use. This qualifier has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of a compiled configuration precludes direct reading of the alias file from any location.

58.11.4.3 **-alternate_recipient=address**

(New in MS 8.0) Specify an alternate recipient address.

58.11.4.4 **-applicationinfo=string**

This qualifier is used to specify the application-info string to use during, for instance, `FROM_ACCESS`, `ORIG_MAIL_ACCESS`, and `MAIL_ACCESS mapping table` probes. For instance, for an incoming SMTP message where the sending client claimed (on its HELO/EHLO line) a hostname of `domain.com`, and where TLS was not used, the application-info string would be `"SMTP/domain.com"`.

58.11.4.5 **-channel (default), -nochannel**

This qualifier controls whether the utility outputs detailed information, *e.g.*, channel flags, regarding the channel an address matches.

58.11.4.6 **-check_expansions, -nocheck_expansions (default)**

This qualifier controls checking of alias address expansion. Normally the MTA considers the expansion of an alias to have been "successful" if *any* of the addresses to which the alias expands are legal. The `-check_expansions` qualifier causes a much stricter policy to be applied: `imsimta test -rewrite -check_expansions` checks each expanded address in detail and reports a list of any addresses, expanded or otherwise, that fail to rewrite properly. For addresses that match the L channel, the MTA also performs validity checks.

58.11.4.7 **-configuration_file=filename**

If no compiled configuration is being used, then `imsimta test rewrite` normally consults the default MTA configuration file during the rewriting process. Prior to MS 7.0, the MTA configuration file was located via the `imta_config_file` option of the MTA Tailor file, usually pointing to `/opt/SUNWmsgsr/config/imta.cnf`; as of MS 7.0, the MTA configuration file is located at `SERVERROOT/config/imta.cnf`. The `-configuration_file` qualifier specifies an alternate file to use in place of the regular configuration file. This switch has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of a compiled configuration will preclude direct reading of the MTA configuration file from any location.

58.11.4.8 **-conversion_file=filename, -noconversion_file**

If no compiled configuration is being used, then `imsimta test rewrite` normally accesses the default conversion file as part of its initialization during the rewriting process; while

the conversion file has no particular effect on address rewriting, it is considered part of the core configuration (and hence `imsimta test -rewrite` will warn of problems accessing the conversion file, or of out-of-date versions of the conversion file). Prior to MS 7.0, that default conversion file was located via the `imta_conversion_file` option of the Tailor file, so usually `/opt/SUNWmsgsr/config/conversions`; as of MS 7.0, the conversion file is located as `SERVERROOT/config/conversions`. The `-conversion_file` qualifier specifies an alternate file to use in place of the regular conversion file. These switches have no effect unless `-noimage_file` is specified or no compiled configuration exists; use of a compiled configuration will preclude direct reading of the conversion file from any location. Use of the `-noconversion_file` qualifier will prevent the conversion file from being read in when there is no compiled configuration.

58.11.4.9 `-database=database-list`

`imsimta test -rewrite` by default during its operation consults any of the usual MTA databases that it has been configured to use. (For MTA configuration controlling whether databases are normally used, see the [Database MTA options](#), and in particular the [alias_magic](#), [use_alias_database](#), [use_domain_database](#), [use_forward_database](#), and [use_reverse_database](#) MTA options.) The `-database` switch is used to either disable references to various databases or to redirect the database paths to nonstandard locations. The allowed list items are `alias`, `noalias`, `personal_alias`, `nopersonal_alias`, `domain`, `nodomain`, `forward`, `noforward`, `general`, `nogeneral`, `reverse`, and `noreverse`. The list items beginning with "no" disable use of the corresponding database. The remaining items require an associated value, which is taken to be the name of that database.

58.11.4.10 `-debug`, `-nodebug` (default)

The address rewriting process is capable of producing additional, detailed explanations of what actions are taken and why. The `-debug` qualifier enables this output; it is disabled by default. In cases of problems with address expansion, `-debug`, especially `-debug=level=3`, can also give more details as to the exact nature of the problem; for instance, the exact LDAP directory error returned in response to a domain lookup up, the exact LDAP directory error returned in response to a user lookup, rejection resulting from an address-based `*_ACCESS` mapping table, *etc.*. As of MS 7.0, note that the basic `-debug` output will report if address "duplicate elimination" occurs, via a debug output line of the form:

```
time-stamp:          - Duplicates previous recipient address, merge
```

This may be of particular interest when multiple, comma-separated addresses were provided initially.

58.11.4.11 `-delivery_receipt`, `-nodelivery_receipt`

The `-delivery_receipt` and `-nodelivery_receipt` qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

58.11.4.12 `-destination_channel=channel`

The `-destination_channel` qualifier controls for which destination or target channel `imsimta test -rewrite` rewrites addresses. Some address rewriting is destination channel specific; this qualifier allows control of the assumed destination channel.

58.11.4.13 `-esmtpushed`, `-noesmtpushed` (default)

(New in JES MS 6.3.) The `-esmtpushed` switch may be used to set an internal flag indicating that ESMTP is in use. In particular, this may be useful when testing `*_ACCESS` mapping tables that make use of that flag (the `$.E` flag).

58.11.4.14 `-expandlimit=n`

(New in JES MS 6.3p1.) This switch may be used to initialize the MTA's internal alias expansion limit; this is intended for testing expansion limit interactions with mailing lists and other MTA facilities.

58.11.4.14.1 `-filter`, `-nofilter` (default)

The `-filter` qualifier may be used to have `imsimta test -rewrite` output any Sieve filters (personal mailbox, so-called "Head of Household", channel, or `system`) applicable for the address in question. Note that a user's `mailAutoReply*` attributes are converted by the MTA into a Sieve "vacation" action which is incorporated at the beginning of the user's personal Sieve filter. As of JES MS 6.1, the filter output will also be labelled as to which Sieve filter it came from, taking the form (under the addresses under the "Submitted address list:" portion of the output):

```
Filter: <type> name location [addr response-addr] [owner owner] (h) [i] {j}
```

or as of JES MS 6.2:

```
Filter: <type> name location [addr response-addr] [owner owner] (h) [i] )k( {j}
```

where `type` is either `system` or `user`, `location` is a URL to the location of the Sieve filter (or in the case of the `system filter`, as of the JES MS 6.2 patch time frame says merely `system:`), `response-addr` is the address to which a notification would be sent back (usually only relevant and non-null for the case of `vacation actions`, in which case it is the envelope From address), and `owner` is the address of the "owner" of this Sieve filter: normally the `local postmaster address` for system-level (the `system` and `channel`) filters, or the user himself for a personal Sieve filter, or the `specified owner` for a "Head of Household" Sieve filter. (The three or four, depending upon version, integers in hexadecimal notation following are internal debug information, showing the location in memory of internal parts of the Sieve structure.) So for instance, one might see output such as the following (where additional line breaks have been inserted for display purposes):

Submitted address list:

```
ims-ms
uid%hosteddomain1.com@ims-ms-daemon (orig first.last@hosteddomain1.com,
inter first.last@hosteddomain1.com, host ims-ms-daemon)
*NOTIFY-FAILURES* *NOTIFY-DELAYS*
Filter: <user> name user:uid%hosteddomain1.com@ims-ms-daemon
[addr uid%hosteddomain1.com@ims-ms-daemon]
[owner uid%hosteddomain1.com@ims-ms-daemon] (0x032107f8) [0x0322b718]
)0x0322ae58( {0x03218e98}
...sample filter lines...
```

```

Filter: <system> name file:///IMTA_TABLE%3Aims-ms.filter
      [addr ] [owner postmaster@host.domain.com]
      (0x00073af8) [0x0009bb28] )0x0009b2c0( {0x0009e670}
....sample filter lines...
Filter: <system> name system: [addr ] [owner postmaster@host.domain.com]
      (0x00d37158) [0x0320de88] )0x0320e788( {0x031ec708}
      header:2000116;0 3 1 :matches 1 "Subject" 1 "ID *... t
      hanks" if 8 ; refuse:2000127;0 1 1 "I think you've se
      nt me a virus.%0AMessage rejected on this basis." ; "" stop
      ;

```

Note that the `-filter` output shows applicable Sieve filters, that is, which Sieve filters *would* get evaluated for this address. But it does not show the actual evaluation of those Sieve filters (as such evaluation can only be done in the context of actual message processing), thus it does not show what the effect(s) of those Sieve filters would be.

58.11.4.14.2 `-from=address, -nofrom`

The `-from` qualifier controls what envelope From address is used for access control probes and mailing list access probes. If this qualifier is omitted, then any such probes use the postmaster return address (as set via the [return_address](#) MTA option). Specifying `-nofrom` tells the MTA to use an empty envelope From address for access probes.

As of MS 7.0.5, note that [returnenvelope](#) (or [mailfromdnsverify](#)) or [return_envelope](#) settings that cause the MTA to attempt a "verification" of the From address can affect `imsimta test -rewrite` output: the output will include a warning if the From address appeared to be problematic, though the input address will still be rewritten as usual.

58.11.4.14.3 `-header, -noheader (default)`

The `-header` qualifier causes the utility to output any applicable [header trimming option file](#) associated with the channel of the destination address. This information will appear after the destination address itself, before any dumped filter information (from the `-filter` qualifier), hence before the "Submitted notifications list:" output. `-noheader` is the default. Note that `-header` merely outputs the header trimming option file; to investigate the potential effect(s) of a header trimming option file, see instead `test -header`.

58.11.4.14.4 `-identifiers, -noidentifiers (default)`

(New in MS 7.0.5) The `-identifiers` switch tells the utility to rewrite as a message identifier, rather than as an address. (For instance, in the default mode of addresses, an `a@b@c` form will be turned into a %-route form, whereas in `-identifiers` mode a value of `a@b@c` would be quoted.)

58.11.4.14.5 `-image_file (default), -noimage_file`

When the `-image_file` switch is specified (the default), `imsimta test -rewrite` will load the compiled configuration. Prior to MS 7.0, this compiled configuration was located via the [imta_config_data](#) option in the [MTA tailor file](#), usually pointing to `/opt/SUNWmsgsr/config/advanced/config_data`. As of MS 7.0, the compiled configuration is located as `SERVERROOT/config/advanced/config_data`. When `-noimage_file` is specified, `imsimta test -rewrite` unconditionally ignores any previously compiled configuration and instead reads configuration information directly from the various text files.

58.11.4.14.6 **-input=input-file-spec**

By default, `imsimta test -rewrite` takes input from `stdin`. The `-input` qualifier may be used to specify a different source for input.

58.11.4.14.7 **-lmtused, -nolmtused (default)**

(New in JES MS 6.3.) The `-lmtused` switch may be used to set an internal flag indicating that LMTP is in use. In particular, this may be useful when testing `*_ACCESS` mapping tables that make use of that flag (the `$:L` flag).

58.11.4.14.8 **-local_alias=value, -nolocal_alias (default)**

This qualifier controls the setting of an alias for the local host. The MTA supports multiple "identities" for the local host; the local host may have a different identity on each channel. This qualifier may be used to set the local host alias to the specified value; appearances of the local host in rewritten addresses will be replaced by this value.

58.11.4.14.9 **-mapping_file[=file-spec], -nomapping_file**

If no compiled configuration is being used, then this qualifier instructs `imsimta test -rewrite` to use the specified mapping file rather than the default mapping file. Prior to MS 7.0, the default mapping file was located via the `imta_mapping_file` option in the [MTA tailor file](#), so usually `/opt/SUNWmsgsr/config/mappings`; as of MS 7.0, the mappings file is located as `SERVERROOT/config/mappings`. These qualifiers have no effect unless `-noimage_file` was specified or no compiled configuration exists; use of any compiled configuration will preclude direct reading of the mappings file. Use of the `-nomapping_file` qualifier will prevent the MTA mapping file from being read in when there is no compiled configuration.

58.11.4.14.10 **-mtpriority=n, -nomtpriority (default)**

(New in MS 8.0) `-mtpriority` takes a required integer argument specifying the initial MTPRIORITY value.

58.11.4.14.11 **-option_file[= filename], -nooption_file**

If no compiled configuration is being used, then the `-option_file` qualifier instructs `imsimta test -rewrite` to use the specified option file rather than the default location MTA option file. Prior to MS 7.0, the MTA option file was located via the `imta_option_file` option in the [MTA Tailor file](#), so usually `/opt/SUNWmsgsr/config/option.dat`; as of MS 7.0, the MTA option file is located at `SERVERROOT/config/option.dat`. These qualifiers have no effect unless `-noimage_file` is specified or no compiled configuration exists; use of any compiled configuration will preclude direct reading of the MTA option file from any location. Use of the `-nooption_file` qualifier will prevent the MTA option file from being read in when there is no compiled configuration.

58.11.4.14.12 **-output=output_file_spec**

By default, `imsimta test -rewrite` writes output to `stdout`. The `-output` qualifier may be used to direct the output of `imsimta test -rewrite` elsewhere.

58.11.4.14.13 **-password=string**

Used to specify the password for a [password-protected list](#).

58.11.4.14.14 **-proxyused, -noproxyused (default)**

(New in JES MS 6.3.) The `-proxyused` switch may be used to set an internal flag indicating that proxy authentication (POP-before-SMTP) is in use. In particular, this may be useful when testing `*_ACCESS` mapping tables that make use of that flag (the `$.P` flag).

58.11.4.14.15 **-read_receipt, -noread_receipt**

The `-read_receipt` and `-noread_receipt` qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

58.11.4.14.16 **-reprocessing (default), -noreprocessing**

By default, the test `-rewrite` utility runs as if the `-reprocessing` switch is set, meaning that some operations that would normally be deferred for "off-line" execution by the [reprocess channel](#) will instead be performed directly by the test address processing. `-noreprocessing` may be used to tell the utility to run in a mode more similar to that of a "normal" channel, where various operations (e.g., [mailing list password lookups](#)) will not be performed by the channel, and where instead the message will be forcibly routed to the `reprocess` channel which is expected to perform the necessary tasks later ("off-line").

58.11.4.14.17 **-restricted=setting**

This qualifier controls the setting of the restricted flag. By default, this flag has value 0. When set to 1, `-restricted=1`, the restricted flag will be set on and addresses will be rewritten using the restricted mailbox encoding format recommended by [RFC 1137](#). This flag is used to force rewriting of address mailbox names in accordance with the RFC 1137 specifications; see the [restricted](#) channel option for further details.

58.11.4.14.18 **-saslused, -nosaslused (default)**

(New in JES MS 6.2p8.) The `-saslused` switch may be used to set an internal flag indicating that SASL authentication (SMTP AUTH) is in use. In particular, this may be useful when testing `*_ACCESS` mapping tables that make use of that flag (the `$.A` flag).

58.11.4.14.19 **-sender=address, -nosender (default)**

New in JES MS 6.2. The `-sender` qualifier may be used to set the "authenticated sender" field, for use in [FROM_ACCESS mapping table](#) probes.

58.11.4.14.20 **-size=n, -nosize (default)**

(New in MS 7.0.5) The `-size` switch sets an assumed "message size" (in bytes), as if the SMTP SIZE extension had been used. It requires an integer argument, which the MTA interprets as being in units of bytes. This can be useful for checking message size-based restrictions. `-nosize` is the default.

58.11.4.14.21 **-soptin, -nosoptin (default)**

(New in MS 7.0 update 2.) Control whether or not to show per-recipient [spamfilter "opt-in"](#).

58.11.4.14.22 **-source_channel=channel**

The `-source_channel` qualifier controls which source channel to assume when rewriting addresses. Some address rewriting is source channel specific; `imsimta test-rewrite` by

default assumes that the channel source for which it is rewriting is the local channel, 1 on UNIX.

58.11.4.14.23 **-spares, -nosparses (default)**

(New in MS 7.0 update 2.) Control whether or not to show per-recipient "spare" LDAP attributes.

58.11.4.14.24 **-statistics, -nostatistics (default)**

The `-statistics` qualifier can be used to tell the MTA to output the direct LDAP lookup cache statistics for the rewrite rule performed; statistics for the domain cache, reverse cache, and alias cache will be displayed. `-nostatistics` is the default.

58.11.4.14.25 **-system_filter=filename, -nosystem_filter**

If no compiled configuration is being used, then `imsimta test -rewrite` normally consults the default system Sieve filter during the rewriting process: this is the `systemfilter` MTA option in Unified Configuration, or the system filter file in legacy configuration. Prior to MS 7.0, the MTA system filter file was located via the `imta_system_filter_file` option of the [MTA Tailor file](#), so usually `/opt/SUNWmsgsr/config/imta.filter`; as of MS 7.0, the MTA system filter file is located as `SERVERROOT/config/imta.filter`. The `-system_filter` qualifier specifies an alternate file to use in place of the default system Sieve filter file. These switches have no effect unless `-noimage_file` is specified or no compiled configuration exists; use of a compiled configuration will preclude direct reading of the system filter file from any location. Use of the `-nosystem_filter` qualifier will prevent the MTA system filter file (legacy configuration) from being read in when there is no compiled configuration.

58.11.4.14.26 **-tag=tag-list, -notag (default)**

(New in MS 7.0.5) The `-tag` switch can be used to set the [conversion tag](#) (or comma-separated list of tags) that will be available at the time of [REVERSE mapping table](#) probes. This can be useful when bit 8 (value 256) of the `include_conversiontag` MTA option is set, so that REVERSE mapping table probes include conversion tags.

58.11.4.14.27 **-tlsused, -notlsused (default)**

(New in JES MS 6.2p8.) The `-tlsused` switch may be used to set an internal flag indicating that TLS is in use. In particular, this may be useful when testing `*_ACCESS` mapping tables that make use of that flag (the `$.T` flag).

58.11.4.14.28 **-transportinfo=string**

This switch is used to specify the transport-info string to use during, for instance, `FROM_ACCESS`, `ORIG_MAIL_ACCESS`, and `MAIL_ACCESS` mapping table probes. (Note that the `PORT_ACCESS` mapping table is *not* checked by the `test -rewrite` utility, as the `PORT_ACCESS` mapping table is consulted for decisions regarding TCP/IP connections, rather than for address handling; in particular the `PORT_ACCESS` mapping table is used by the [Dispatcher](#), and then again by SMTP server processes, at a much earlier stage of processing than the address rewriting process.) Note that a typical transport-info string, of the form

TCP | *server-address* | *server-port* | *client-address* | *client-port*

contains vertical bar characters, |, which will require some quoting to pass through the shell; e.g.,

```
-transportinfo=TCP\|123.45.67.8\|12435\|10.0.0.1\|25
```

58.11.4.14.29 -xml_config[=file-path]

(New in MS 7.0.) `imsimta test -rewrite` normally reads its configuration from `IMTA_TABLE:config.xml`, if such a file exists. The `-xml_config` switch specifies use of a Unified Configuration (which is the MTA's default behavior if `IMTA_TABLE:config.xml` exists), and optionally specifies an alternate, XML format, configuration file to use in place of `IMTA_TABLE:config.xml`.

58.11.5 Examples

This UNIX example shows typical output generated by `imsimta test -rewrite` in MS 6.3. Perhaps the single most important piece of information generated by `imsimta test -rewrite` is displayed on the last few lines of the output, (6), which shows the channel to which `imsimta test -rewrite` would submit a message with the specified test address and the form in which the test address would be rewritten for that channel. This output is invaluable when debugging configuration problems.

```
% imsimta test -rewrite dan@innosoft.com
channel                = tcp_local                (1)
channel description    =
channel caption        =
channel user filter    =
dest channel filter    =
source channel filter  =
channel flags #0       = BIDIRECTIONAL SINGLE_SYSTEM IMMORMAL NOSERVICEALL (
channel flags #1       = SMTP_CRLF MX IDENTNONENUMERIC DEFAULT
channel flags #2       = COPYSENDPOST COPYWARNPOST POSTHEADBODY HEADERINC NOE
channel flags #3       = LOGGING NORESTRICTED RETAINSECURITYMULTIPARTS
channel flags #4       = EIGHTNEGOTIATE HEADERKEEPORDER NOHEADERREAD RULES
channel flags #5       = TRUNCATESMTPLONGLINES
channel flags #6       = LOCALUSER REPORTNOTARY
channel flags #7       = SWITCHCHANNEL REMOTEHOST DATEFOUR DAYOFWEEK
channel flags #8       = NODEFRAGMENT EXQUOTA REVERSE NOCONVERT_OCTET_STREAM
channel flags #9       = NOTHURMAN INTERPRETENCODING USEINTERMEDIATE RECEIVED
default host          = domain.com domain.com
linelength            = 998
addrspersfile         = 99
channel env addr type  = SOURCEROUTE
channel hdr addr type  = SOURCEROUTE
channel official host  = tcp-daemon                (3)
channel queue 0 name   = SMTP_POOL
channel queue 1 name   = SMTP_POOL
channel queue 2 name   = SMTP_POOL
channel queue 3 name   = SMTP_POOL
channel after params   =
channel user name      =
```

```

urgentnotices      = 1 2 4 7
normalnotices     = 1 2 4 7
nonurgentnotices  = 1 2 4 7
channel rightslist ids =                                     (4)
local behavior flags = %x0
expandchannel     =
notificationchannel =
dispositionchannel =
tlsswitchchannel  =
backward channel  = tcp_local                               (5)
unique identifier  = dan@innosoft.com
header forward address = dan@innosoft.com (route (TCP-DAEMON,TCP-DAEMON)) (host in
header reverse address = dan@innosoft.com
envelope forw address = dan@innosoft.com (route (TCP-DAEMON,TCP-DAEMON)) (host in
envelope rev address  = dan@innosoft.com (route (TCP-DAEMON,TCP-DAEMON)) (host in
name               =
mbox               = dan
Extracted address action list:
    dan@innosoft.com
Extracted 733 address action list:
    dan@innosoft.com
Address list expansion:
-13 expansion total.
Expanded address:
    dan@innosoft.com
Submitted address list:                               (6)
    tcp_local
    dan@innosoft.com (orig dan@innosoft.com, host innosoft.com) *NOTIFY-FAILURES* *N

Submitted notifications list:                         (7)

```

1. The channel to which, after rewriting as an envelope To address, the address is mapped.
2. The flags set for the channel indicated in (1). These flags are controlled by the channel options on the first line of the channel control block for the specified channel. Any unknown options---options which may have been mistyped---will be interpreted as group ids and will appear on the line (4).
3. The channel's official host name as specified on the second line of the channel control block for the channel indicated in (1).
4. Any items appearing on the first line of the channel block which were not channel options are interpreted as group ids. Any group ids so specified for the channel are listed on this line.
5. The channel which the address would match if rewritten as an envelope From address.
6. The channel to which a message with the address dan@innosoft.com would be queued and the envelope To address which would be used. Here, the message would be submitted to

the TCP/IP channel, `tcp_local`, using the address `dan@innosoft.com`. Other information appearing here might include an explicit `Errors-to:` address, which, if present, appears enclosed in square brackets; or notations such as `*RR*` or `*NRR*`, indicating whether or not the message is flagged for read receipts, or notations such as `*NOTIFY FAILURES*`, `*NOTIFY DELAYS*`, `*NOTIFY SUCCESSES*`, *etc.*, indicating the message's delivery receipt mechanism and flagging.

7. Notification addresses. If notifications need to be generated regarding this address, as for instance in the case of a group or list whose definition includes some (immediately obvious as such) bad addresses, then the addresses about which a notification needs to be generated will be listed here, along with the error corresponding to each such address. If an override envelope `From` is in effect for the original message, hence if the notification will go back to some address other than the original message's sender, then that address (the address to which the notification will be sent) will be shown enclosed in square brackets. Note that the recipient address for the notification will only be shown if it is something different than the original sender address (as specified via the `-from` qualifier, or defaulting to the postmaster address). New in Ancho, the word "to" will appear within such square brackets, to emphasize that the address shown is the address to which the notification will be sent.

58.12 version utility

Print MTA version number.

58.12.1 Syntax

```
imsimta version
```

58.12.2 restrictions

Must have superuser privileges.

58.12.3 Parameters

None.

58.12.4 Description

`imsimta version` prints out the MTA version number, and displays the system's name, operating system release number and version, and hardware type.

58.12.5 Example

```
# imsimta version
Oracle Communications Messaging Server 7.0.5.33.0 64bit (built Aug 22 2014)
libimta.so 7.0.5.33.0 64bit (built 00:04:52, Aug 22 2014)
Using /opt/sun/comms/messaging64/config/config.xml (compiled)
NSS Library Version: 3.16.3 Basic ECC
SunOS example 5.10 Generic_127128-11 i86pc i386 i86pc
```

The above example shows MTA version information for a Solaris x86 system running the 64 bit version of Oracle Messaging Server 7.0u6, running with a compiled unified configuration.

Part VII Additional components

Messaging Server includes a [Personal Addressbook facility](#), [SNMP support](#), and support for an [Event Notification Service](#).



Chapter 59 PAB options

59.1 enable Option Under <code>pab</code>	59-1
59.2 <code>defaulthostindex</code> Option	59-1
59.3 <code>active</code> Option	59-1
59.4 <code>alwaysusedefaulthost</code> Option	59-1
59.5 <code>attributelist</code> Option	59-1
59.6 <code>ldapbasedn</code> Option	59-1
59.7 <code>ldapbinddn</code> Option	59-2
59.8 <code>ldaphost</code> Option	59-2
59.9 <code>ldappasswd</code> Option	59-2
59.10 <code>ldapport</code> Option	59-2
59.11 <code>ldapussessl</code> Option	59-2
59.12 <code>maxnumberofentries</code> Option	59-2
59.13 <code>migrate415</code> Option	59-2
59.14 <code>numberofhosts</code> Option	59-2

A number of options affect PAB (Personal Address Book) operation.

Note that for purposes of MTA queries of PABs, there are a number of [PAB-related MTA options](#) that can affect the MTA's PAB queries.

59.1 enable Option Under `pab`

The `enable` PAB option enables or disables the Personal Address Book (PAB) feature.

59.2 `defaulthostindex` Option

The `defaulthostindex` PAB option specifies the index of the default host.

59.3 `active` Option

The `active` PAB option should be set to 1 if PAB host is active, 0 otherwise.

59.4 `alwaysusedefaulthost` Option

The `alwaysusedefaulthost` PAB option enables one PAB server to be used (overriding hostname in PAB URIs).

59.5 `attributelist` Option

The `attributelist` PAB option allows adding new attributes to a personal address book entry. With this parameter, you can create an attribute that does not already exist.

59.6 `ldapbasedn` Option

The `ldapbasedn` PAB option specifies the base DN for PAB searches. If not set, it defaults to the value of the [ugldapbasednbase option](#) (`local.ugldapbasedn` in legacy configuration).

59.7 ldapbinddn Option

The `ldapbinddn` PAB option specifies the bind DN for PAB searches. If not set, it defaults to the value of the [ugldapbinddnbase option](#) (`local.ugldapbinddn` in legacy configuration).

59.8 ldaphost Option

The `ldaphost` PAB option specifies the hostname of the PAB Directory Server. If not set, it defaults to the value of the [ugldaphostbase option](#) (`local.ugldaphost` in legacy configuration).

59.9 ldappasswd Option

The `ldappasswd` PAB option specifies the password for the user specified by the [ldapbinddn](#) PAB option (respectively, `local.service.pab.ldappasswd` and `local.service.pab.ldapbinddn` in legacy configuration). If not set, `ldappasswd` defaults to the value of the [ugldapbindcredbase option](#) (`local.ugldapbindcred` in legacy configuration).

59.10 ldapport Option

The `ldapport` PAB option specifies the port number of the PAB Directory Server. If not set, it defaults to the value of the [ugldapportbase option](#) (`local.ugldapport` in legacy configuration).

59.11 ldapusessl Option

The `ldapusessl` PAB option specifies whether to use SSL to connect to the PAB Directory Server.

59.12 maxnumberofentries Option

The `maxnumberofentries` PAB option specifies the maximum number of entries a single PAB can store.

59.13 migrate415 Option

The `migrate415` PAB option enables PAB migration when set to 1.

59.14 numberofhosts Option

The `numberofhosts` PAB option specifies the number of PAB servers (up to a maximum of 16).

Chapter 60 SNMP options

60.1 enable Option Under snmp	60-1
60.2 listenaddr Option Under snmp	60-1
60.3 port Option Under snmp	60-1
60.4 cachettl Option	60-1
60.5 contextname Option	60-2
60.6 directoryscan Option	60-2
60.7 enablecontextname Option	60-2
60.8 registerindices Option	60-3
60.9 servertimeout Option	60-3
60.10 standalone Option	60-3

Several options affect operation of Messaging Server's [SNMP subagent\(s\)](#). Some options only affect Messaging Server's Net-SNMP based SNMP subagent, used on Solaris platforms running Solaris 10 and later, as well as Linux platforms, but do *not* apply to the legacy SNMP subagent supplied for Solaris platforms running Solaris 9 and earlier operating systems.

See also the [logfile options](#) set as `snmp.logfile.*`.

60.1 enable Option Under snmp

The `enable` SNMP option, `snmp.enable` (Unified Configuration) or `local.snmp.enable` (legacy configuration), enables the SNMP subagent on `start-msg` startup.

This option applies to both the Net-SNMP based SNMP subagent, and the legacy SNMP subagent for Solaris 9 and earlier.

60.2 listenaddr Option Under snmp

The `listenaddr` SNMP option specifies the IPv4 address to listen on when running as a SNMP master agent -- that is, when the [standalone](#) SNMP option has been set to 1. The allowed values for `listenaddr` include an IPv4 address in dotted decimal form (e.g., 127.0.0.1), or a short form or fully-qualified DNS host name which will be resolved to an IPv4 address by obtaining the DNS A record for the name. To explicitly specify the default value of binding to all available interfaces, the string "INADDR_ANY" may be used. To bind to the loopback device, 127.0.0.1, the string "localhost" may be specified.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

60.3 port Option Under snmp

The `port` SNMP option specifies the UDP port to listen on when running as a SNMP master agent -- that is, when the [standalone](#) SNMP option has been set to 1.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

60.4 cachettl Option

The `cachettl` SNMP option specifies the time to live (TTL) in seconds for cached monitoring data. That is, 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; (see also the [directoryscan](#) SNMP option).

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.

This option applies to both the Net-SNMP based SNMP subagent, and the legacy SNMP subagent for Solaris 9 and earlier.

60.5 contextname Option

When the use of SNMP v3 context names has been enabled with the [enablecontextname](#) SNMP option, then the `contextname` SNMP option may be used to explicitly set the context name used by the subagent for its MIBs. The value supplied for this option is a string value and must be appropriate for use as a SNMP v3 context name. If not explicitly set, the default value for `contextname` is the value of the [defaultdomain](#) base option (the `service.defaultdomain configutil` parameter in legacy configuration). The `contextname` option is ignored when `enablecontextname` has the value 0.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

60.6 directoryscan Option

The `directoryscan` SNMP option selects whether or not the subagent performs scans of the on-disk channel message queues to count `.HELD` message files, and update its information on the oldest message files. That information corresponds to the `mtaGroupLoopsDetected`, `mtaGroupOldestMessageStored` and `mtaGroupOldestMessageId` MIB variables. When this option has the value 1 (or "true" in legacy configuration), then a cache of this information is maintained and updated as needed. Sites with thousands of queued messages, if not interested in these particular MIB variables, should consider setting this option's value to 0, as performing the message queue traversal involves some overhead.

This option applies to both the Net-SNMP based SNMP subagent, and the legacy SNMP subagent for Solaris 9 and earlier.

60.7 enablecontextname Option

The Net-SNMP based SNMP subagent has the ability to register its MIBs under an SNMP v3 context name. 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).

The `enablecontextname` SNMP option controls whether to register this instance's MIBs under an SNMPv3 context name. When the option is set to a value of 1, the subagent will default to using the value of `defaultdomain` for its context name, unless a different context name has been specified using via the `contextname` SNMP option.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

60.8 registerindices Option

The `registerindices` SNMP option controls whether to register as visible MIB variables the `applIndex`, `assocIndex`, and `mtaGroupIndex` MIB indices. By default, these MIB variables are implicit and not explicitly shown as distinct MIB variables when walking the MIBs.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

60.9 servertimeout Option

The `servertimeout` SNMP option specifies the maximum number of seconds to wait for each step in probing a server (connect to, read from, write to, etc.).

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.

This option applies to both the Net-SNMP based SNMP subagent, and the legacy SNMP subagent for Solaris 9 and earlier.

60.10 standalone Option

The `standalone` SNMP option may be set to run as a standalone SNMP agent when set to 1 (or "true" for the legacy configuration `local.snmp.standalone configutil` parameter).

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 the `standalone` option a value of 0 (or "false" for the legacy configuration parameter). However, 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 `snmp.listenaddr` and `snmp.port` options (configutil parameters `local.snmp.listenaddr` and `local.snmp.port` in legacy configuration). To run in this standalone mode, specify a value of 1 (or "true" in legacy configuration) for this option.

Running in standalone mode does not interfere with other SNMP master or subagents running on the system.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

Chapter 61 ENS options

61.1 <code>enable</code> Option Under <code>ens</code>	61-1
61.2 <code>port</code> Option Under <code>ens</code>	61-1
61.3 <code>enablsslport</code> Option Under <code>ens</code>	61-1
61.4 <code>sslport</code> Option Under <code>ens</code>	61-1
61.5 <code>secret</code> Option Under <code>ens</code>	61-2
61.6 <code>loglevel</code> Option Under <code>ens</code>	61-2
61.7 <code>domainallowed</code> Option Under <code>ens</code>	61-2
61.8 <code>domainnotallowed</code> Option Under <code>ens</code>	61-2
61.9 <code>sslnicknames</code> Option Under <code>ens</code>	61-2

Several options affecting operation of the ENS server may be set under the `ens` group; in particular, `ens.enable` must be set to enable running the ENS server. For additional, logging related options, see the [logfile options](#) set as `ens.logfile.*`.

For related functionality and options, see also the [base.listenaddr](#) option, the [enssub value of the debugkeys](#) option, the `msprobe.probe:ens.*`, and the [notifytarget options](#).

61.1 `enable` Option Under `ens`

The `enable` ENS option enables the ENS server on `start-msg` startup. The default value is the value of `store.enable` (aka `local.store.enable` in legacy config).

61.2 `port` Option Under `ens`

The `port` ENS option specifies the TCP port the ENS server will listen on. Versions of ENS prior to 7 update 4 support inclusion of a server IP address in addition to a port number in the `local.ens.port configutil` option. That syntax is deprecated. In legacy configuration, the IP address portion is ignored as of 8.0.1. In Unified Configuration, the legacy syntax is disallowed. The [base.listenaddr](#) option (`service.listenaddr` in legacy configuration) is the recommend way to set the ENS server host IP for version 7 update 4 and later.

If `ens.port` is set, then the [notifytarget:target-name.ensport](#) option (`local.store.notifyplugin.*.ensport` in legacy configuration) must be configured to match. In legacy configuration, if the `local.ens.port configutil` parameter also specified a host IP address, then `local.store.notifyplugin.*.enshost` would also need to be set correspondingly. For Unified Configuration, the equivalent is that the [notifytarget:target-name.enshost](#) option value must match the [base.listenaddr](#) option value.

61.3 `enablsslport` Option Under `ens`

The `enablsslport` ENS option sets whether or not ENS over SSL service is started.

61.4 `sslport` Option Under `ens`

The `sslport` ENS option specifies the port number for the ENS over SSL port. Note that to enable the ENS+SSL service, the [ens.enablsslport](#) option must be set.

61.5 secret Option Under ens

The `secret` ENS option specifies the secret used to authenticate ENS clients with the server.

61.6 loglevel Option Under ens

The `loglevel` ENS option specifies the level of ENS client library logging to the process `nslog` file. Messages are also filtered per the `loglevel` of the process. Allowed values are as in `logfile.*.loglevel`. But the value "debug" generates lots of data and is not recommended.

61.7 domainallowed Option Under ens

The `domainallowed` ENS option specifies [access filters](#) specifying which domains and/or IP addresses are allowed ENS access.

61.8 domainnotallowed Option Under ens

The `domainnotallowed` ENS option specifies [access filters](#) specifying which domains and/or IP addresses are not allowed ENS access.

61.9 sslnicknames Option Under ens

The `sslnicknames` ENS option specifies a list of SSL/TLS server certificate nicknames (only one per certificate type) for HTTP to offer clients if SSL/TLS enabled. Overrides for HTTP the base level `sslnicknames` option (corresponding to the legacy configuration `encryption.rsa.nssslpersonalityssl configutil` parameter).

Chapter 62 eval_ldapd options

62.1 domainallowed Option Under eval_ldapd	62-1
62.2 domainnotallowed Option Under eval_ldapd	62-1

The Evaluation LDAP server (eval_ldapd) supports options controlling access via [access filters](#).

62.1 domainallowed Option Under eval_ldapd

The domainallowed option under eval_ldapd allows setting [access filters](#) specifying which domains and/or IP addresses are allowed evaluation ldapd access.

62.2 domainnotallowed Option Under eval_ldapd

The domainnotallowed option under eval_ldapd allows setting [access filters](#) specifying which domains and/or IP addresses are *not* allowed evaluation ldapd access.

Appendix A Supported Standards

This information lists national, international, industry, and de-facto standards related to electronic messaging and for which support is claimed by Oracle Communications Messaging Server. Most of these are Internet standards, published by the [RFC Editor](#) and approved by the [Internet Engineering Task Force \(IETF\)](#). Standards for documents from other sources are noted.

Software compliant with older IETF RFCs is usually compatible with newer RFCs unless the protocol version changes. Where the table mentions a newer version in parentheses and that version is not listed in a separate row, we have not made a complete assessment of compatibility but are probably compatible with the new version unless otherwise stated. Newer versions of IETF specifications often drop features that were optional, rarely used or problematic. If we implemented features of note that are present only in the older specification, the older specification is listed on a separate row in addition to the parenthetical mention.

Features with status "Experimental" or "Work in Progress" may be changed incompatibly in a future version until the specification has settled. Features with status "De Facto" may not interoperate with all software as interpretations of the de facto standard may vary.

A.1 Protocols

Reference	Status	Feature and Information
RFC 3501	Proposed standard	<i>IMAP</i> (port 143) (Usage: Server Client) (Older specs: RFC2060) March 2003
RFC 2180	Informational	<i>IMAP4 Multi-Accessed Mailbox Practice</i> (port 143) (Usage: Server Client) July 1997
RFC 2061	Informational	<i>IMAP4 Compatibility with IMAP2bis</i> (port 143) (Usage: Server Client) (Older specs: RFC1730) December 1996
RFC 2062	Informational	<i>Internet Message Access Protocol - Obsolete Syntax</i> (port 143) (Usage: Server Client) December 1996
RFC 2683	Informational	<i>IMAP4 Implementation Recommendations</i> (port 143) (Usage: Server Client) September 1999
RFC 1730	Proposed standard	<i>IMAP4</i> (Usage: Server) (Newer specs: RFC2060 RFC2061) December 1994
RFC 3501 section 7.2.1	Proposed standard	<i>IMAP4rev1</i> (Usage: Server Client) (Older specs: RFC2060) March 2003

RFC 3501 section 6.2.1	Proposed standard	<i>IMAP STARTTLS</i> (Usage: Server Off-by-default) (Older specs: RFC2060) March 2003
RFC 2246	Proposed standard	<i>TLS 1.0 as used by IMAP STARTTLS</i> (Usage: Server Off-by-default) (Newer specs: RFC4346) January 1999
RFC 3501 section 6.2.3	Proposed standard	<i>IMAP LOGINDISABLED</i> (Usage: Server Off-by-default) (Older specs: RFC2060) March 2003
RFC 4422	Proposed standard	<i>AUTHENTICATE (SASL)</i> Caveat: SASLprep is not implemented. (Usage: Server Client) (Older specs: RFC2222) June 2006
RFC 3501 section 6.2.2	Proposed standard	<i>AUTHENTICATE (SASL)</i> Caveat: SASLprep is not implemented. (Usage: Server Client) (Older specs: RFC2060) March 2003
RFC 4616	Proposed standard	<i>IMAP AUTH=PLAIN</i> Caveat: SASLprep is not implemented. (Usage: Server Client) August 2006
RFC 4422 page 29	Proposed standard	<i>IMAP AUTH=EXTERNAL</i> (Usage: Server Off-by-default) (Older specs: RFC2222) June 2006
RFC 2245	Proposed standard	<i>IMAP AUTH=ANONYMOUS</i> (Usage: Server Off-by-default) (Newer specs: RFC4505) November 1997
RFC 2195	Proposed standard	<i>IMAP AUTH=CRAM-MD5</i> (Usage: Server Off-by-default) (Older specs: RFC2095) September 1997
RFC 2086	Proposed standard	<i>IMAP ACL</i> Caveat: Support for the 'p' right only works for user 'anyone' presently. (Usage: Server) (Newer specs: RFC4314) January 1997
RFC 2087	Proposed standard	<i>IMAP QUOTA</i> (Usage: Server) January 1997
RFC 2088	Proposed standard	<i>IMAP LITERAL+</i> (Usage: Server) January 1997
RFC 2342	Proposed standard	<i>IMAP NAMESPACE</i> (Usage: Server) May 1998

RFC 2359	Proposed standard	<i>IMAP UIDPLUS</i> (Usage: Server) (Newer specs: RFC4315) June 1998
RFC 3348	Informational	<i>IMAP CHILDREN</i> (Usage: Server) July 2002
RFC 3516	Proposed standard	<i>IMAP BINARY</i> (Usage: Server) April 2003
RFC 3691	Proposed standard	<i>IMAP UNSELECT</i> Added in release: 6.3 (Usage: Server) February 2004
RFC 2177	Proposed standard	<i>IMAP IDLE</i> Added in release: 6.3 (Usage: Server Off-by-default) (Enhancement Request: 12037435) June 1997
RFC 4959	Proposed standard	<i>IMAP SASL-IR</i> Added in release: 7.0 (Usage: Server) September 2007
RFC 4469	Proposed standard	<i>IMAP CATENATE</i> Added in release: 7.0 (Usage: Server) April 2006
RFC 5161	Proposed standard	<i>IMAP ENABLE</i> Added in release: 7.0 (Usage: Server) March 2008
RFC 4731	Proposed standard	<i>IMAP ESEARCH</i> Added in release: 7.0 (Usage: Server) November 2006
RFC 4551	Proposed standard	<i>IMAP CONDSTORE</i> Added in release: 7.0 (Usage: Server) (Newer specs: RFC7162) June 2006
RFC 5092	Proposed standard	<i>IMAP URL</i> Details: Used by IMAP referrals, URLFETCH and CATENATE Added in release: 8.0.1 (Usage: Server Client) (Older specs: RFC2192) (Enhancement Request: 21092120) November 2007
RFC 4467	Proposed standard	<i>IMAP URLAUTH</i> Added in release: 7.0 (Usage: Server Client) May 2006
RFC 5162	Proposed standard	<i>IMAP QRESYNC</i> Added in release: 7.0 (Usage: Server) (Newer specs: RFC7162) March 2008
RFC 5032	Proposed standard	<i>IMAP WITHIN</i> Added in release: 7.0 (Usage: Server) September 2007

RFC 2221	Proposed standard	<i>IMAP Login Referrals</i> Details: Used to detect user in transit via rehostuser utility. Added in release: 7.0 (Usage: Server Client) October 1997
RFC 5257	Experimental	<i>IMAP ANNOTATE</i> Added in release: 7.0 (Usage: Server) June 2008
RFC 5267	Proposed standard	<i>IMAP CONTEXT</i> Added in release: 7.0 (Usage: Server) July 2008
RFC 5256	Proposed standard	<i>IMAP THREAD</i> Caveat: We don't implement I18NLEVEL or i;unicode-casemap as required, and instead use a similar unicode collator Added in release: 7.0 (Usage: Server) June 2008
RFC 5256	Proposed standard	<i>IMAP SORT</i> Caveat: We don't implement I18NLEVEL or i;unicode-casemap as required, and instead use a similar unicode collator Added in release: 6.3 (Usage: Server Client) June 2008
RFC 5957	Proposed standard	<i>IMAP SORT=DISPLAY</i> Caveat: We don't implement I18NLEVEL or i;unicode-casemap as required, and instead use a similar unicode collator Added in release: 8.0 (Usage: Server) (Enhancement Request: 19597156) July 2010
RFC 5255	Proposed standard	<i>IMAP LANGUAGE</i> (Usage: Server Client) June 2008
RFC 5464	Proposed standard	<i>IMAP METADATA</i> Added in release: 7.0.5 (Usage: Server) February 2009
RFC 2971	Proposed standard	<i>IMAP ID</i> Added in release: 7 Update 4 (Usage: Server) (Enhancement Request: 12139651) October 2000
RFC 5530	Proposed standard	<i>IMAP Response Codes</i> Added in release: 7.0.5 (Usage: Server) May 2009
RFC 4314	Proposed standard	<i>IMAP ACL Extension</i> (Usage: Server) (Older specs: RFC2086) December 2005

RFC 5182	Proposed standard	<i>IMAP Referencing the Last SEARCH Result</i> Added in release: 7 Update 2 (Usage: Server) March 2008
RFC 5819	Proposed standard	<i>IMAP4 Extension for Returning STATUS Information in Extended LIST</i> Added in release: 8.0 (Usage: Server) (Enhancement Request: 16773851) March 2010
RFC 6154	Proposed standard	<i>IMAP LIST Extension for Special-Use Mailboxes</i> Added in release: 8.0 (Usage: Server) (Enhancement Request: 16773916) March 2011
RFC 7377	Proposed standard	<i>IMAP MULTISEARCH</i> Added in release: 8.0 (Usage: Server) (Older specs: RFC6237) (Enhancement Request: 17596568) October 2014
RFC 2221	Proposed standard	<i>IMAP LOGIN-REFERRALS</i> Added in release: 8.0.1 (Usage: Server Client) (Enhancement Request: 21092120) October 1997
RFC 1939	Internet standard	<i>POP3</i> (port 110) (Usage: Server Client) (Older specs: RFC1725) May 1996
RFC 1957	Informational	<i>Some Observations on Implementations of the Post Office Protocol (POP3)</i> (port 110) (Usage: Server Client) June 1996
RFC 1939 section 7	Internet standard	<i>POP TOP</i> (Usage: Server) (Older specs: RFC1725) May 1996
RFC 1939 page 12	Internet standard	<i>POP UIDL</i> (Usage: Server Client) (Older specs: RFC1725) May 1996
RFC 1939 page 13	Internet standard	<i>POP USER / PASS</i> (Usage: Server Client) (Older specs: RFC1725) May 1996
RFC 1939 page 15	Internet standard	<i>APOP</i> (Usage: Server Off-by-default) (Older specs: RFC1725) May 1996
RFC 2449 section 6.3	Proposed standard	<i>POP SASL</i> (Usage: Server Client) November 1998
RFC 2449	Proposed standard	<i>POP CAPA</i> Caveat: We don't implement EXPIRE

		or LOGIN-DELAY. (Usage: Server) November 1998
RFC 2449 section 6.6	Proposed standard	<i>POP PIPELINING</i> (Usage: Server) November 1998
RFC 2449 section 6.4	Proposed standard	<i>POP RESP-CODES</i> (Usage: Server) November 1998
RFC 2449 section 6.9	Proposed standard	<i>POP IMPLEMENTATION</i> (Usage: Server) November 1998
RFC 2595 section 4	Proposed standard	<i>POP STLS</i> (Usage: Server Off-by-default) June 1999
RFC 2246	Proposed standard	<i>TLS 1.0 as used by POP STLS</i> (Usage: Server Off-by-default) (Newer specs: RFC4346) January 1999
RFC 3206	Proposed standard	<i>POP AUTH-RESP-CODES</i> (Usage: Server) February 2002
RFC 2821	Proposed standard	<i>SMTP Specification</i> (port 25) (Usage: Server Client) (Older specs: RFC0821 RFC0974 RFC1869) (Newer specs: RFC5321) April 2001
RFC 821	Internet standard	<i>SMTP Legacy Specification</i> (port 25) (Usage: Server Client) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 920	Unknown	<i>Domain requirements</i> (port 25) (Usage: Server Client) October 1984
RFC 974	Historic	<i>SMTP Mail Routing and DNS</i> (port 25) (Usage: Server Client) (Newer specs: RFC2821) January 1986
RFC 1123 section 5	Internet standard	<i>SMTP Host Requirements</i> (port 25) (Usage: Server Client) October 1989
RFC 2505	Best current practice	<i>Anti-Spam Recommendations for SMTP MTAs</i> (port 25) (Usage: Server Client) February 1999
RFC 3848	Draft standard	<i>ESMTP and LMTP Transmission Types Registration</i> (port 25) (Usage: Server Client) July 2004
RFC 1428	Informational	<i>Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME</i> (port 25) (Usage: Server Client) February 1993

RFC 6152	Internet standard	<i>SMTP 8BITMIME</i> (Usage: Server Client) (Older specs: RFC1652) March 2011
RFC 2920	Internet standard	<i>SMTP PIPELINING</i> (Usage: Server Client) (Older specs: RFC2197) September 2000
RFC 3030	Proposed standard	<i>SMTP CHUNKING</i> Caveat: Spec also contains BINARYMIME which has not been implemented. Added in release: 6.3 (Usage: Server Client) (Older specs: RFC1830) (Enhancement Request: 12143903) December 2000
RFC 3461	Draft standard	<i>SMTP DSN</i> (Usage: Server Client) (Older specs: RFC1891) January 2003
RFC 2034	Proposed standard	<i>SMTP ENHANCEDSTATUSCODES</i> (Usage: Server Client) October 1996
RFC 3463	Draft standard	<i>Enhanced Mail System Status Codes</i> (Usage: Server Client) (Older specs: RFC1893) January 2003
RFC 2821 section 4.1.1.7	Proposed standard	<i>SMTP EXPN</i> (Usage: Server) (Older specs: RFC0821 RFC0974 RFC1869) (Newer specs: RFC5321) April 2001
RFC 2821 section 4.1.1.8	Proposed standard	<i>SMTP HELP</i> (Usage: Server) (Older specs: RFC0821 RFC0974 RFC1869) (Newer specs: RFC5321) April 2001
RFC 821 section 3.4	Internet standard	<i>SMTP SAML</i> (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.4	Internet standard	<i>SMTP SEND</i> (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.4	Internet standard	<i>SMTP SOML</i> (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.8	Internet standard	<i>SMTP TURN</i> (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982

RFC 3207	Proposed standard	<i>SMTP STARTTLS</i> (Usage: Server Client Off-by-default) (Older specs: RFC2487) February 2002
RFC 2246	Proposed standard	<i>TLS 1.0 as used by SMTP STARTTLS</i> (Usage: Server Client Off-by-default) (Newer specs: RFC4346) January 1999
RFC 4954	Proposed standard	<i>SMTP AUTH</i> Caveat: SASLprep is not implemented. Versions 8.0 and prior may use different error codes than those in RFC 4954. Support for AUTH=mail parameter is server-only. (Usage: Server Client Off-by-default) (Older specs: RFC2554) July 2007
RFC 4422	Proposed standard	<i>SMTP AUTH</i> Caveat: SASLprep is not implemented. Versions 8.0 and prior may use different error codes than those in RFC 4954. Support for AUTH=mail parameter is server-only. (Usage: Server Client Off-by-default) (Older specs: RFC2222) June 2006
RFC 1985	Proposed standard	<i>SMTP ETRN</i> (Usage: Server) August 1996
RFC 1870	Internet standard	<i>SMTP SIZE</i> (Usage: Server) (Older specs: RFC1653) November 1995
RFC 1846	Experimental	<i>SMTP 521 Reply Code</i> (Usage: Client) September 1995
draft-ietf-appsawg-nullmx-07.txt	Work in Progress, subject to change	<i>Null MX No Service Resource Record</i> Details: Used to provide more timely reporting of SMTP errors Added in release: 8.0 (Usage: Client)
RFC 7372	Proposed standard	<i>Email Authentication Status Codes</i> Details: Used to provide more accurate reporting of SMTP DNS errors Added in release: 8.0 (Usage: Client) September 2014
RFC 3865	Proposed standard	<i>SMTP NO-SOLICITING</i> (Usage: Server Off-by-default) September 2004

SMTP AUTH=LOGIN	Widely Used or De Facto	<i>SMTP AUTH=LOGIN</i> Details: Undocumented pre-standard subset of AUTH PLAIN but with interoperability problems. (Usage: Server)
RFC 4408	Experimental	<i>Sender Permitted From</i> (Usage: Client) (Newer specs: RFC7208) April 2006
RFC 2852	Proposed standard	<i>SMTP Deliver By</i> Details: Control when message is returned as undeliverable Added in release: 8.0.1 (Usage: Server Client) June 2000
draft-melnikov-smtp-altrecip-on-error-01.txt	Work in Progress, subject to change	<i>SMTP ALTRECIP</i> Details: Delivery to alternate recipient on error Added in release: 8.0.1 (Usage: Client)
RFC 6710	Proposed standard	<i>SMTP Priority</i> Details: Envelope specification of message processing priority Added in release: 8.0 (Usage: Server Client) August 2012
RFC 6710	Proposed standard	<i>SMTP Priority Tunneling</i> Details: Tunneling of SMTP Message Transfer Priorities Added in release: 8.0.1 (Usage: Server Client) August 2012
RFC 6729	Proposed standard	<i>Received state</i> Details: Received: field indicator of the state attached to the message Added in release: 8.0 (Usage: Server) September 2012
RFC 7293	Proposed standard	<i>Require Recipient Valid Since</i> Details: Mailbox ownership change detection Added in release: 8.0 (Usage: Server Client) July 2014
RFC 6409	Internet standard	<i>Message Submission</i> Details: Superset of SMTP adding AUTH by default. (port 587) (Usage: Server Client) (Older specs: RFC4409) November 2011
RFC 4954	Proposed standard	<i>Submission AUTH</i> Caveat: SASLprep is not implemented. Versions 8.0 and prior may use different error codes than those in RFC 4954. Support for AUTH=mail parameter is server-only.

RFC 4422	Proposed standard	(Usage: Server Client) (Older specs: RFC2554) July 2007 <i>Submission AUTH</i> Caveat: SASLprep is not implemented. Versions 8.0 and prior may use different error codes than those in RFC 4954. Support for AUTH=mail parameter is server-only. (Usage: Server Client) (Older specs: RFC2222) June 2006
RFC 4616	Proposed standard	<i>Submission AUTH PLAIN</i> Caveat: SASLprep is not implemented. (Usage: Server Client) August 2006
RFC 4422 page 29	Proposed standard	<i>Submission AUTH=EXTERNAL</i> (Usage: Server Client Off-by-default) (Older specs: RFC2222) June 2006
Submission AUTH=LOGIN	Widely Used or De Facto	<i>Submission AUTH=LOGIN</i> Details: Undocumented, subset of AUTH PLAIN functionality. (Usage: Server Client)
RFC 4468	Proposed standard	<i>Submission BURL</i> Added in release: 7.0 (Usage: Server) May 2006
RFC 4865	Proposed standard	<i>Submission Future Release</i> Added in release: 7.0 (Usage: Server) May 2007
RFC 3458	Proposed standard	<i>Message Context for Internet Mail</i> Details: Gateway for pager/cell phone connectivity for SMS. (Usage: Server Client Off-by-default) January 2003
RFC 2251	Proposed standard	<i>LDAPv3</i> (port 389) (Usage: Client) (Newer specs: RFC4510 RFC4511 RFC4513 RFC4512) December 1997
RFC 1034	Internet standard	<i>Domain names - concepts and facilities</i> (port 53) (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035	Internet standard	<i>Domain names - implementation and specification</i> (port 53) (Usage: Client) (Older specs:

		RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.4.1	Internet standard	<i>DNS A</i> (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 3596	Draft standard	<i>DNS AAAA</i> Details: Support can be configured on (Usage: Client) (Older specs: RFC3152 RFC1886) October 2003
RFC 1035 section 3.3.1	Internet standard	<i>DNS CNAME</i> (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.3.9	Internet standard	<i>DNS MX</i> (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 934	Unknown	<i>DNS MX</i> (Usage: Client) January 1985
RFC 1035 section 3.3.12	Internet standard	<i>DNS PTR</i> (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.3.14	Internet standard	<i>DNS TXT</i> Details: For RBL and SPF. (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 3507	Informational	<i>ICAP</i> Details: For Norton AV. (port 1344) (Usage: Client Off-by-default) April 2003
RFC 1928	Proposed standard	<i>SOCKS 5</i> (port 1080) (Usage: Client Off-by-default) March 1996
RFC 2788	Proposed standard	<i>SNMP Network Services MIB</i> Details: Provided via NetSNMP. (port 161) (Usage: Server Off-by-default) (Older specs: RFC2248) March 2000
RFC 2789	Proposed standard	<i>SNMP Mail MIB</i> Details: Provided via NetSNMP. (port 161) (Usage: Server Off-by-default) (Older specs: RFC2249 RFC1566) March 2000
RFC 2033	Informational	<i>LMTP</i> Caveat: The addressing model is non-standard in order to avoid the need for the LMTP server to perform

address lookups in LDAP.
Third-party use of this is not supported and will not work as expected. (port 225) (Usage: Server Client Off-by-default) October 1996

A.2 Data and File Formats, Schema

Reference	Status	Feature and Information
RFC 822	Internet standard	<i>Legacy Internet Message Format</i> (Usage: Generate Accept Process) (Older specs: RFC0733) (Newer specs: RFC2822) August 1982
RFC 5322	Draft standard	<i>Internet Message Format</i> (Usage: Generate Accept Process) (Older specs: RFC2822) October 2008
RFC 2045	Draft standard	<i>Multi-purpose Internet Mail Extensions (MIME)</i> (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2046	Draft standard	<i>Multi-purpose Internet Mail Extensions (MIME)</i> (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2049	Draft standard	<i>Multi-purpose Internet Mail Extensions (MIME)</i> (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2047	Draft standard	<i>MIME International Headers</i> (Usage: Generate Accept Process Normalize) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2231	Proposed standard	<i>MIME International Parameters</i> (Usage: Accept Process) (Older specs: RFC2184) November 1997
RFC 2183	Proposed standard	<i>MIME Content-Disposition</i> (Usage: Generate Accept Process) (Older specs: RFC1806) August 1997

RFC 1864	Draft standard	<i>Content-MD5</i> (Usage: Process Validate) (Older specs: RFC1544) October 1995
RFC 3458	Proposed standard	<i>Message Context for Internet Mail</i> Details: For per-context quotas. Added in release: 6.3 (Usage: Generate Accept) January 2003
RFC 1740	Proposed standard	<i>Macintosh MIME Format</i> (Usage: MTA Translate Off-by-default) December 1994
RFC 1741	Informational	<i>Macintosh Binhex Encoding</i> (Usage: MTA Translate Off-by-default) December 1994
RFC 2045 section 6.7	Draft standard	<i>MIME base-64 and quoted-printable encodings</i> (Usage: Generate Accept Process MTA Translate Validate Normalize) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 1847	Proposed standard	<i>MIME multipart/signed</i> (Usage: Process) October 1995
RFC 2480	Proposed standard	<i>Gateways and MIME Security Multiparts</i> (Usage: Process) January 1999
RFC 3280	Proposed standard	<i>X.509 Certificate Profile</i> Details: Generated by certutil. Accepted by SSL/TLS. (Usage: Accept Off-by-default) (Older specs: RFC2459) (Newer specs: RFC5280) April 2002
RFC 3462	Draft standard	<i>MIME multipart/report</i> (Usage: Generate Process) (Older specs: RFC1892) (Newer specs: RFC6522) January 2003
RFC 3464	Draft standard	<i>Delivery Status Notifications (DSN)</i> (Usage: Generate Process) (Older specs: RFC1894) January 2003
RFC 3798	Draft standard	<i>Message Disposition Notifications (MDN)</i> (Usage: Generate Process Off-by-default) (Older specs: RFC2298) May 2004
RFC 2442	Informational	<i>Batch SMTP</i> (Usage: Generate Accept Off-by-default) November 1998

RFC 5228	Proposed standard	<i>Sieve (Mail Filtering Language)</i> (Usage: Accept) (Older specs: RFC3028) January 2008
RFC 5173	Proposed standard	<i>Sieve body</i> Details: Includes some storage and complexity limits. Added in release: 7 Update 3 (Usage: Accept Off-by-default) April 2008
RFC 3894	Proposed standard	<i>Sieve :copy parameter</i> (Usage: Accept) October 2004
RFC 5260	Proposed standard	<i>Sieve date and index</i> Added in release: 7 Update 4 (Usage: Accept) (Enhancement Request: 12183152) July 2008
RFC 7352	Proposed standard	<i>Sieve duplicate</i> Added in release: 8.0 (Usage: Accept) September 2014
RFC 5293	Proposed standard	<i>Sieve Editheader</i> Added in release: 6.3 (Usage: Accept) August 2008
RFC 5228	Proposed standard	<i>Sieve Encoded-character extension</i> Added in release: 7.0 (Usage: Accept) (Older specs: RFC3028) January 2008
RFC 6069	Experimental	<i>Sieve envelope-dsn, redirect-dsn</i> Details: Messaging Server Support for envelope DSN access. Added in release: 7 Update 2 (Usage: Accept) December 2010
RFC 5183	Proposed standard	<i>Sieve Environment</i> Added in release: 7 Update 1 (Usage: Accept) May 2008
RFC 5429	Proposed standard	<i>Sieve ereject</i> Added in release: 7.0 (Usage: Accept) (Older specs: RFC3028) (Enhancement Request: 12228618) March 2009
draft-ietf-sieve-external-lists-10.txt	Work in Progress, subject to change	Messaging Server Support for Externally Stored Lists Sieve Extension Details: Likely subject to incompatible change in a future version. Caveat: Implementation includes additional features not in the current specification. Added in release: 7 Update 1 (Usage: Accept Off-by-default)

RFC 5463	Proposed standard	<i>Sieve ihave</i> Added in release: 7.0 (Usage: Accept) March 2009
RFC 5232	Proposed standard	<i>Sieve imap4flags</i> Added in release: future (Usage: Accept) (Enhancement Request: 12108510) January 2008
RFC 5703	Proposed standard	<i>Sieve MIME extension</i> Caveat: Only mime and foreverypart implemented, other extensions in spec not implemented (foreverypart added in pimento release) Added in release: 7 Update 2 (Usage: Accept) October 2009
draft-martin-sieve-notify-01.txt	Work in Progress, subject to change	<i>Sieve Notifications by email</i> Caveat: We only implement notify method email. Added in release: future (Usage: Accept Off-by-default) (Enhancement Request: 12070010)
RFC 5435	Proposed standard	<i>Sieve extension for notifications</i> Caveat: Only mailto: URLs are presently supported Added in release: 7.0.5 (Usage: Accept) January 2009
RFC 5436	Proposed standard	<i>Sieve notification mechanism: mailto</i> Added in release: 7.0.5 (Usage: Accept) January 2009
RFC 5231	Proposed standard	<i>Sieve Relational Tests</i> (Usage: Accept) (Older specs: RFC3431) January 2008
RFC 5233	Proposed standard	<i>Sieve subaddress</i> Added in release: 6.0 (Usage: Accept) (Older specs: RFC3598) January 2008
RFC 5235	Proposed standard	<i>Sieve Spamtest / Virustest</i> (Usage: Accept) (Older specs: RFC3685) January 2008
RFC 5230	Proposed standard	<i>Sieve vacation</i> (Usage: Accept) January 2008
RFC 6131	Proposed standard	<i>Sieve vacation seconds</i> Added in release: 7.0.5 (Usage: Accept) July 2011
RFC 5229	Proposed standard	<i>Sieve variables</i> Added in release: 6.3 (Usage: Accept) January 2008

RFC 952 section 1	Unknown	<i>Domain Name Syntax</i> (Usage: Accept Validate) (Older specs: RFC0810) October 1985
RFC 1123 section 2.1	Internet standard	<i>Domain Name Syntax</i> (Usage: Accept Validate) October 1989
RFC 2251	Proposed standard	<i>namingContexts of root DSE</i> Details: Used by comm_dssetup.pl with Directory Server 6 to determine naming contexts so customer can select one to use for user/group tree. Added in release: 6.3 (Usage: Accept) (Newer specs: RFC4510 RFC4511 RFC4513 RFC4512) December 1997
RFC 2849	Proposed standard	<i>LDAP Data Interchange Format (LDIF)</i> (Usage: Generate) June 2000
RFC 2254	Proposed standard	<i>LDAP Search Filter String Format</i> (Usage: Generate Accept Validate) (Older specs: RFC1960) (Newer specs: RFC4510 RFC4515) December 1997
RFC 2253	Proposed standard	<i>LDAP Distinguished Name String Format</i> (Usage: Generate MTA Translate Normalize) (Older specs: RFC1779) (Newer specs: RFC4510 RFC4514) December 1997
RFC 2255	Proposed standard	<i>LDAP URL Format</i> (Usage: Accept Validate) (Older specs: RFC1959) (Newer specs: RFC4510 RFC4516) December 1997
RFC 3280	Proposed standard	<i>PKIX</i> Details: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile Caveat: As implemented by Mozilla NSS library, believed to be complete. Some new features in RFC 5280 not implemented. (Usage: Generate Accept Validate) (Older specs: RFC2459) (Newer specs: RFC5280) April 2002

ITU-T Rec. X.680, ITU-T Rec. X.690.	Other National/International	<i>ASN.1</i> Caveat: Only the LDAP, PKIX and SNMP subsets are implemented. (Usage: Generate Accept Validate)
RFC 1738 section 3.10	Proposed standard	<i>File URL Format</i> (Usage: Accept) (Newer specs: RFC4248 RFC4266) December 1994
RFC 3339	Proposed standard	<i>Internet Date / Time Timestamps</i> Details: Also ISO 8601. Used in a few places that don't require mail-specific timestamp format. (Usage: Generate) July 2002
Unix /var/mail mailbox file format	Widely Used or De Facto	<i>Unix /var/mail mailbox file format</i> Details: Traditional 'From ' Unix mailbox format (multiple messages per file). Used by MTA native channel and migration tool. (Usage: Generate MTA Translate)
RFC 4627	Informational	<i>application/json</i> Details: Presently only used internally as well as between WMAP and Convergence. Caveat: Unicode escapes in JSON strings are not implemented in internal C JSON library. (Usage: Generate Accept) (Newer specs: RFC7159) July 2006

A.3 Charsets

Reference	Status	Feature and Information
ANSI X3.4-1986	Other National/International	<i>US-ASCII</i> Details: ANSI X3.4-1986 (Usage: Accept MTA Translate Validate IMAP Search)
RFC 3629	Internet standard	<i>UTF-8</i> Details: Unicode Transformation Format 8-bit. (Usage: Accept MTA Translate Validate Normalize IMAP Search) (Older specs: RFC2279) November 2003
ISO 8859 (1-10,14,15)	National/International	<i>ISO 8859 (1-10,14,15)</i> Details: ISO 8859 family of 8-bit charsets. Several languages with proper US-ASCII subset.

			(Usage: MTA Translate IMAP Search)
ISO 8859 11	National/International		<i>ISO 8859 11</i> Details: ISO 8859 8-bit Thai with ASCII subset. Added in release: 7 Update 3 (Usage: MTA Translate IMAP Search)
CNS 11643-1986, GB 2312-1980	Other National/International		<i>GB-2312</i> Details: Simplified Chinese, also CNS 11643-1986. (Usage: MTA Translate IMAP Search)
RFC 1842	Informational		<i>HZ-GB-2312</i> Details: Simplified Chinese. No MTA support. (Usage: IMAP Search) August 1995
Big5			<i>Big5</i> Details: Traditional Chinese (Usage: MTA Translate IMAP Search)
GB-18030	National/International		<i>GB-18030</i> Details: Extended Simplified Chinese with Unicode subset. (Usage: MTA Translate) (Enhancement Request: 12068067)
ISO/IEC 2022-1986	Other National/International		<i>ISO-2022</i> Details: ISO standard technique for code-switching; very complex so everything uses a subset. Caveat: We only implement a subset of full 2022 as needed by language-specific charsets and other MTA functions. (Usage: MTA Translate IMAP Search)
RFC 1468	Informational		<i>ISO-2022-JP</i> Details: Japanese, based on ISO 2022, JIS X 0201-1976, JIS X 0208-1990 (Usage: MTA Translate IMAP Search) June 1993
RFC 1554	Informational		<i>ISO-2022-JP-2</i> Details: Japanese, with additional language support, not widely used. (Usage: MTA Translate IMAP Search) December 1993
JIS X 0201-1976, JIS X 0208-1990 with EUC encoding	Other National/International		<i>EUC-JP</i> Details: Japanese. (Usage: MTA Translate IMAP Search)
RFC 1557	Informational		<i>ISO-2022-KR</i> Details: Korean KSC 5601-1987 with ISO 2022 encoding (Usage: MTA

		Translate IMAP Search) December 1993
RFC 1557	Informational	<i>EUC-KR</i> Details: Korean KSC 5601-1987 with EUC encoding (Usage: MTA Translate IMAP Search) December 1993
RFC 1345	Informational	<i>ks_c_5601-1987</i> Details: Alternate name for Korean KSC 5601-1987 with EUC encoding Added in release: 7.0.5.34.0 (Usage: MTA Translate IMAP Search) June 1992
RFC 1489	Informational	<i>KOI8-R</i> Details: Russian 8-bit with US-ASCII subset. (Usage: MTA Translate IMAP Search) July 1993
Thai Industrial Standard 620-2533	Other National/International	<i>TIS-620</i> Details: TIS-620. IMAP SEARCH support new in 7u4 (Usage: MTA Translate IMAP Search)
Windows-1250	Vendor Non-standard, subject to change	<i>Windows-1250</i> Details: Windows variant of ISO-8859-2 (Eastern/Central Latin) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
Windows-1251	Vendor Non-standard, subject to change	<i>Windows-1251</i> Details: Windows variant of KOI8-R (Russian) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
Windows-1252	Vendor Non-standard, subject to change	<i>Windows-1252</i> Details: Windows variant of ISO-8859-1 (Latin) with extensions. (Usage: MTA Translate IMAP Search)
Windows-1253	Vendor Non-standard, subject to change	<i>Windows-1253</i> Details: Windows incompatible variant of ISO-8859-7 (Greek) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
Windows-1254	Vendor Non-standard, subject to change	<i>Windows-1254</i> Details: Windows variant of ISO-8859-9 (Turkish) with extensions. IMAP SEARCH

Windows-1255	Vendor Non-standard, subject to change	support new in 7u3 (Usage: MTA Translate IMAP Search) <i>Windows-1255</i> Details: Windows incompatible variant of ISO-8859-8 (Hebrew) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search) (Enhancement Request: 12272449)
Windows-1256	Vendor Non-standard, subject to change	<i>Windows-1256</i> Details: Windows incompatible variant of ISO-8859-6 (Arabic) with extensions. (Usage: MTA Translate IMAP Search)
Windows-1257	Vendor Non-standard, subject to change	<i>Windows-1257</i> Details: Windows incompatible variant of ISO-8859-13 (Baltic) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
Windows-1258	Vendor Non-standard, subject to change	<i>Windows-1258</i> Details: Windows Vietnamese 8-bit. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
RFC 2152	Informational	<i>UTF-7</i> Details: Use not recommended, doesn't interoperate well. (Usage: MTA Translate IMAP Search) (Older specs: RFC1642) May 1997
RFC 3501 section 5.1.3	Proposed standard	<i>IMAP Modified UTF-7</i> (Usage: Accept MTA Translate Validate) (Older specs: RFC2060) March 2003

A.4 Library files

Reference	Status	Feature and Information
RFC 1345	Informational	<i>charnames.txt</i> Details: Defines the mnemonic character names (based on RFC 1345) used in <i>charsets.txt</i> (Usage: Accept) June 1992
RFC 1345	Informational	<i>charsets.txt</i> Details: Character set definitions based on RFC 1345 (Usage: Accept) June 1992

ISO 3166 country codes	Other National/International	<i>countries.txt</i> Details: List of ISO 3166 country codes (Usage: Accept)
IANA top-level domain list	Other National/International	<i>tlds.txt</i> Details: Copy of http://data.iana.org/TLD/tlds-alpha-by-domain.txt (Usage: Accept)
RFC 3066	Best current practice	<i>languages.txt</i> Details: List of language codes (Usage: Accept) (Older specs: RFC1766) (Newer specs: RFC4646 RFC4647) January 2001

Glossary

A

- APOP [RFC 1939 \(POP3\)](#) defines the APOP command. This is an alternate method for authenticating the user which, rather than sending the username and password in the clear over the network, encodes the password.
- Authentication mechanism An authentication mechanism is a particular method for a client to prove its identity to a server. APOP, PLAIN and CRAM-MD5 (mechanism names are as defined by [RFC 2222 \(SASL\)](#)), are examples of authentication mechanisms. Another, but non-standard, mechanism is the LOGIN mechanism. For discussions of particular mechanisms, see for instance [RFC 2195](#) documenting CRAM-MD5, and [RFC 1939](#) documenting APOP.
- Authentication source An authentication source is a file, database, interface to an LDAP directory, *etc.*, which is accessible to the server and wherein are stored authentication verifiers for users. The system password file, the user portion of the DIT in an LDAP directory, or a certificate database, are examples of authentication sources.
- Authentication verifier An authentication verifier (*e.g.*, password) is stored on the server and contains information used to verify a user's identity. The format of the authentication verifier may restrict which mechanisms can be used. The term authentication verifier is preferred in place of password, since while passwords are the most common instance of authentication verifiers, an authentication verifier could also be something like a certificate in an LDAP directory or the like; usually, however, one may think "password" wherever one sees "authentication verifier".

C

- Certificate In the security context, a certificate is a guarantee, signed by some trusted authority, that says that a piece of information is what it purports to be. For instance, certificates are often needed and encountered when using a public key pair.
See Also Public key pair.
- Certificate Authority A Certificate Authority is a recognized, generally well-trusted authority that is willing to sign other organization's certificates. A Certificate Authority has a well-published certificate (containing their public key) that other organizations can use to verify the Certificate Authority's signature on other certificates. Assuming that an organization trusts the Certificate Authority, this then gives them confidence in certificates that include a valid signature from the Certificate Authority. Verisign, Inc. (now owned by Symantec Corp.), and Thawte Consulting are two of the better known commercial Certificate Authorities. Large corporations will sometimes, for their own internal purposes, act as their own Certificate Authority.

Certificate request	A certificate request is a special form of a site's public key suitable for signing by a Certificate Authority. The signing of a certificate request generates a certificate. See Also Certificate Authority.
Channel	In the context of the Oracle MTA, a channel is an abstract combination of message transport and protocol, realized through a channel program. Each different transport and protocol combination has a corresponding different type of MTA channel. A channel represents a connection with another computer system or group of systems, over which messages may be transported. For instance, the MTA supports SMTP-over-TCP/IP channels, for sending and receiving messages from the Internet (as well as internal networks); typically, several distinct SMTP-over-TCP/IP channels are configured, for separate handling of SMTP-over-TCP/IP message traffic originating from, or destined to, different categories of remote and local systems. See Channels . See Also Channel program.
Channel block	The definition of an MTA channel appearing as a named set of options under a <code>channel</code> group (Unified Configuration) or in the MTA configuration file (legacy configuration) is called a channel block; see Channel configuration . See Also Channel.
Channel keyword	In Unified Configuration, channel options take the place of the legacy configuration channel keywords. A large number of channel options (channel keywords in legacy configuration) are available for use in MTA channel definitions (channel blocks) to control and modify the action of the channel to which a keyword is applied; see Channel options . See Also Channel.
Channel program	Loosely speaking, any program which enqueues or dequeues messages to or from the MTA's message queues. See the master_command and slave_command Job Controller options . See Also Channel.
CIDR (Classless Inter-Domain Routing)	CIDR is a method for allocating IP addresses and routing IP packets. In IPv4, it permits allocating address space on any address bit boundary, rather than necessary in 8-bit groups.
CIDR notation	A notation of specifying IP addresses and their associated routing prefix. It appends a slash character to the address and the decimal number of leading bits of the routing prefix. See Also CIDR (Classless Inter-Domain Routing).
Cipher suite	A cipher suite is a set of cryptographic algorithms used for key exchange, encryption, and generation of hashes and signatures on content. In order for the SSL or TLS network protocol to be used, both sides of a communication must support a common cipher suite. Examples of cipher suites include RSA, Diffie-Hellman, Triple DES, and IDEA. See Also SSL (Security Sockets Layer), TLS (Transport Layer Security).
Common name	In X.500 terminology, the common name or <code>commonName</code> or <code>CN</code> attribute is a multi-valued attribute that describes an entry; typically

it is something like a person's name, "First Middle Last", *etc.* The term is also used in other contexts, such as in a certificate, and in non-X.500 directories, especially LDAP directories.
See Also Certificate, X.500.

Conversion tag

Conversion tags are a private-to-the-MTA envelope field in message copies traversing the MTA. Conversion tags are stored per message copy (message file); so cases where different message recipients need to have different conversion tags set require that the MTA generate separate message files. Conversion tags may be initially set due to user or domain LDAP attributes (see [ldap_conversion_tag](#), [ldap_source_conversion_tag](#), [ldap_domain_attr_conversion_tag](#), and [ldap_domain_attr_source_conversion_tag](#)) or set in Sieve scripts (see [Sieve conversiontag extensions](#)) or set in Unified Configuration in an alias option (see [alias_conversion_tag](#)) or correspondingly in legacy configuration in an alias file entry (see the `[CONVERSION_TAG]named parameter`), or set in an [address access mapping table](#) (see the `$G` flag discussed in [Address access mapping table flags](#)). Conversion tag values may optionally be used in various mapping tables (see the [include_conversiontag](#) MTA option)), or may be tested in Sieve scripts (see [Sieve conversiontag extensions](#)). Conversion tags are normally consumed by the [conversion channel](#); see the TAG conversion parameter discussed in Available conversion parameters, listed alphabetically.

CRAM-MD5

[RFC 2195 \(IMAP/POP AUTHorize Extension\)](#) defines the CRAM-MD5 mechanism (Challenge-Response Authentication Mechanism using the MD5 digest algorithm) for authenticating using an encoded password, rather than sending the user's password in the clear over the network.

D

Dequeue

The act of removing a mail message from the MTA's message queues. (Often but not always such removal is due to delivering the message; however dequeue encompasses other cases of removal, such as when a message expires (delivery attempts time out), or is intentionally deleted by the mail system administrator via a utility such as `imsimta qm delete`, or when an original message with multiple recipients has some recipients successfully delivered while other recipients suffer temporary failures, thereby resulting in a dequeue of the original, multi-recipient, message combined with a re-enqueue of a new message copy containing only the not-yet-delivered recipients.)

Directory Information Tree

Information in an LDAP directory is considered to be in the form of a directory information tree: information is stored in nodes, arranged in a hierarchical (tree) structure.

Distinguished name

In X.500 terminology, the distinguished name or distinguishedName or DN attribute uniquely identifies an object in the Directory Information Tree. The term is also used in other contexts, such as in a certificate, and in non-X.500 directories, especially LDAP directories.
See Also Directory Information Tree.

DIT	An abbreviation for Directory Information Tree. See Also Directory Information Tree.
DNS (Domain Name System)	Distributed naming system for computers, services, <i>etc.</i> , on the Internet or on private networks, that in particular translates domain names into numerical IP addresses.
DNSBL	Another term for Realtime Black List (RBL). See Also RBL (Realtime Blackhole List, Realtime Block List, Realtime Blacklist).

E

EAI (Email Address Internationalization)	EAI is a proposal from the EAI IETF working group to permit internationalized email, and in particular internationalized (non-ASCII) email addresses. See RFC 6530 (Overview and Framework for Internationalized Email) for an overview.
encoded-word	In the context of MIME messages and particularly in the headers of messages, see the MIME specifications, especially RFC 2047 . See Also MIME.
Enqueue	The act of submitting for transmission a mail message to the MTA.
Envelope	Additional information may also be present in a message envelope, including notification flags, and various MTA-private fields. Sieve filters may access certain message envelope fields using the " envelope " test.) See Also Message envelope.

F

Final address	The "final" form of a recipient address is intended for machine-delivery, not necessarily user-visibility, and may include delivery-specific information, such as, in the case of messages being delivered to the Message Store, folder names or host names or MTA channel names. This "final" form is thus in contrast to the original form of the address, and the "intermediate" form also used by the MTA. See Also Intermediate address.
FQDN (Fully Qualified Domain Name)	A domain name specified fully, all the way out to a Top Level Domain, <i>e.g.</i> , host.domain.com, not merely a short form hostname. See Also TLD (Top Level Domain).

G

General database	In MTA usage, the term general database refers to a particular database, that can be used by the MTA for a variety of purposes; see for instance rewrite rule general database substitutions and mapping table general database substitutions . Depending upon the setting of the MTA option use_text_databases , the general "database" is either stored and accessed in the form of an on-disk, true database (the default), or else is stored as an in-memory, hashed structure built from an on-disk
------------------	--

file. Alternatively, new in MS 8.0, it may be stored and accessed via memcache; see the [general_database_url](#) MTA option.

Grey-listing

Grey-listing refers to issuing a temporary rejection to attempted submissions of SMTP messages from "new" (not previously encountered) senders; that is, a temporary rejection is issued the first time the sender attempts to send the message, while accepting the message on any later submission attempt. This approach to reducing spam is based on the theory that automated spam blasters typically will not bother to attempt to resend messages, whereas messages from legitimate remote correspondents will get additional delivery attempts performed by remote MTAs. But not all sites are willing to accept imposing a delay on accepting messages from "new" remote senders.

GUI

A Graphics User Interface or GUI is a visually oriented interface, such as typically seen on Mac or Windows systems.

I

ICAP (Internet Content Adaptation Protocol)

See [RFC 3507 \(Internet Content Adaptation Protocol \(ICAP\)\)](#), ICAP provides a lightweight protocol for executing a "remote procedure call" on HTTP messages. As such, it is suitable for purposes such as [spam/virus scanning](#) and [HTML sanitization](#).

IETF

The IETF, Internet Engineering Task Force, is the Internet standards body.

Intermediate address

The MTA commonly deals with several forms of address, including the preserved "original" form of a recipient address (the ORCPT form--- in principle, that form originally typed by the sending user, but in practice that true original form may not have been preserved and instead some "later", transformed version may be the earliest form preserved), the "recently" active form (referred to as the "intermediate" form) corresponding to the form of the address produced after list expansion but prior to any most recent forwarding applied by the MTA, and an altered "final" form of that recipient address (as for instance a final form after forwarding is applied, or after user mailbox name generation occurs as with the "mailbox" delivery option). For instance, the "intermediate" form of an address might be `first.last@hosted.domain.com` in contrast to a "final" form of `uid%hosted.domain.com@ims-ms-daemon`. See Also Final address.

K

Keyword

Shorthand for Channel keyword.
See Also Channel keyword.

L

local-part

The local-part is the non-domain portion of an email address: the portion sometimes thought of as the "username", plus optionally a subaddress; see [RFC 5322](#).

See Also Subaddress.

M

MADMAN	MADMAN was the name of the Mail and Directory Management Working Group. The MADMAN group originated RFC 1566 (Mail Monitoring MIB) .
Mailbox filter	Mailbox filters are rules for individual users, for MTA channels, or for the MTA as a whole, specifying screening and certain delivery handling features for incoming messages. Nowadays, Sieve filters are the common form of mailbox filters (and Sieve filters are also used in some non-MTA contexts, such as post-delivery processing by Sieve-enabled email clients, \ or use by the Message Store in filtering which messages to purge). See Also Sieve.
Mapping table	Many components of the MTA make use of one or another mapping table: a table mapping input strings to output strings. All MTA mapping tables are stored in Unified Configuration as mapping XML elements, or in legacy configuration are stored in the MTA's mapping file.
Mass mailing	A mass mailing is any message sent to a (relatively) large number of recipients. The recipients might be all of the users hosted at a site, all of the users in some domain, all users in some organizational unit, all members (including external members) of a large mailing list, <i>etc.</i> The purpose of the message might be one of great urgency (such as an emergency communication), or it might be of general interest but low urgency (such as announcements). A number of considerations and configuration options relevant to mass mailings are discussed under Mass mailings .
Master channel program	Any program which enqueues messages to the MTA's message queues. See Also Slave channel program.
Message archiving	Message archiving is a term used in several different senses, with different purposes. Two of the most common senses/forms are as follows. (1) Compliance archiving: By this is meant keeping a long term record of all message traffic (or of certain selected message traffic) to comply with legal or other requirements. For such compliance archiving, certainty that "relevant" messages have been captured is usually paramount; but the captured messages are intended for administrative/legal access, not normally for further direct end-user access. (2) Operational archiving: By this is meant "off-loading" storage of certain messages, especially for instance older messages, to more economical storage/access mechanisms. Part of operational archiving is an expectation that end-users will still be able to access "their" messages, when desired, reasonably conveniently. Some additional discussion of message archiving, along with techniques of interest, may be found under Message archival and hashing MTA options .
Message envelope	A message envelope specifies for MTA purposes (message routing purposes) who a message was sent from, and who its recipients are. The message envelope is not normally visible (directly) to end users, other than in special ways such as the Return-path: header line value (which

records the final form of the envelope From: address), or as the list of message recipients in the user's message user agent message composition interface. (End users instead see the message header and message body.) In terms of the SMTP protocol, the message envelope includes the MAIL FROM: and RCPT TO: portions of the SMTP dialogue. The `imsimta qm` utility's `read` command may be used to show these fundamental envelope fields (as well as headers and optionally content) of a message in the MTA's queues. The MTA maintains additional envelope fields in its message files in the MTA queues, which it uses as appropriate and configured. (Note that some such fields have standardized meanings, such as notification flags; others are private to the MTA.) The format of all envelope fields as stored in message files in the MTA queue area is private to the MTA and hence envelope fields should not be accessed directly through inspection of message files, but rather be accessed through normal, supported MTA mechanisms, or if being accessed for custom purposes, should be accessed via the MTA SDK. Examples of normal, supported mechanisms involving envelope fields would include: Sieve filter access to certain message envelope fields using the "envelope" test; the [conversion channel](#) use of [conversion tags](#); [MTA transaction logging](#) that may include envelope data; *etc.*

- Message Store The component of a messaging system that contains the user mailboxes, and facilities (*e.g.*, IMAP, POP, and HTTP servers) to access those messages; *e.g.*, the Oracle [Message Store](#)
- MIB MIB is an acronym for Management Information Base. In the email context, see [RFC 1566 \(Mail Monitoring MIB\)](#).
- MIME MIME stands for Multipurpose Internet Mail Extensions. It is the format used for standard representation of email messages across the Internet (and also used in HTTP for the World Wide Web). It was originally specified in [RFC 1521](#) and [RFC 1522](#), later updated by RFCs 2045--2049.
- Moderated mailing list mailing list where only some senders are allowed to post directly to the list, while attempted postings from other senders are re-routed to a list moderator or moderators, who can then approve and post the message to the list, or disapprove the message and thereby prevent it from being posted.
- MTA Message transfer agent; *e.g.*, the Oracle Messaging Server [MTA](#).
- MUA Mail user agent; see UA.
See Also UA (User Agent).

N

- NOTARY See originally RFCs 1891-1894, now obsoleted by RFCs 3461-3464.

P

- Polling In the e-mail context, polling tends to mean an active step to query whether there are messages or data for the MTA to receive. This is as opposed to the MTA passively waiting to receive messages or data. For

instance, the SMTP TURN and ESMTP ETRN commands are examples of polling at the SMTP protocol level.

Private key

A private key is the secret half of a public key pair.
See Also Public key pair.

Public key

A public key is the published half of a public key pair.
See Also Public key pair.

Public key encryption

So-called public key encryption refers to encryption and decryption using a pair of keys, referred to as a public key pair. One key is referred to as the public key, and is generally published (visible to the outside world); the other key is referred to as the private key and is secret and known only to the owner of the public key pair. User A may encrypt data to send to user B using user B's public key, and then only user B will be able to decrypt the data by using user B's own private key.
See Also Public key pair, Public key, Private key.

Public key pair

Public key encryption uses pairs of keys, one kept secret and one published (made accessible) to the outside world. What the public key encrypts, the private key decrypts, and *vice-versa*. The keys together are called a public key pair.

R

RBL (Realtime Blackhole List, Realtime Block List, Realtime Blacklist)

An RBL is a list, usually published through the DNS, of the IP addresses of problem message senders (spammers). The MTA can be configured to consult such lists via any of several mechanisms, most notably a [dns_verify callout](#).

Rewrite rules

Also called [domain rewriting rules](#): those rules specified in the MTA as [rewrite XML elements](#) in Unified Configuration, or configured in the upper half of the legacy configuration file, for transforming domain names in addresses.

RFC

Request For Comments; the Internet's method of publishing documents.

RFC 822 address

[RFC 822 \(Standard for the format of ARPA Internet text messages\)](#) defined the format for Internet e-mail addresses.
See Also EAI (Email Address Internationalization).

S

SASL (Simple Authentication and Security Layer)

See [RFC 2222 \(Simply Authentication and Security Layer\)](#).

Schema

Definitions, including structure and syntax, of the types of information that can be stored as entries in an LDAP Directory Server.

Schema tag

A tag (name) identifying the schema in use. The Oracle Messaging Server MTA has native support for several schemata (and can potentially support other, more or less compatible schemata), such as the Netscape

	<p>Messaging Server 4.1 schema (schema tag <code>nms41</code>), the Sun Internet Mail Server 4.0 schema (schema tag <code>sims40</code>), and the iPlanet Messaging Server 5.0 schema (schema tag <code>ims50</code>); the MTA merely needs to be told which schema is in use. See in particular the <code>ldap_schematag</code> MTA option (and in legacy configuration, also see the <code>configutil</code> parameter <code>local.imta.schematag</code>).</p> <p>See Also Schema.</p>
Security rule set	<p>A security rule set is a set of rules determining which authentication mechanisms and sources are permitted or used by a server. For the MTA, the <code>PORT_ACCESS</code> mapping table is used to determine the security rule set to apply to an incoming connection, based on IP addresses and ports.</p>
Sieve	<p>In the context of e-mail, the term "Sieve" usually refers to the Sieve mail filtering language originally defined in RFC 3028, and extended in additional RFCs including RFC 3431 (Sieve Extension: Relational Tests), RFC 3598 (Sieve Email Filtering---Subaddress Extensions), RFC 3685 (Sieve Email Filtering: Spamtest and VirusTest Extensions), RFC 3894 (Sieve Extension: Copying Without Side Effects). Sieve scripts written in the Sieve filtering language can specify tests to perform on incoming messages, and then actions to take based on the tests, such as discarding certain messages, or filing certain messages into specific folders, etc. The Messaging Server MTA supports Sieve scripts, both at the system level (an MTA-wide Sieve script, and channel level scripts), and user Sieve scripts stored and evaluated at the MTA level (stored either in LDAP or in files on the MTA server).</p>
Sign	<p>In the context of a TLS certificate, to say that a certificate is signed means that a Certificate Authority has generated a hash of the contents of the certificate and used their own private key to encrypt that hash and then appended that encrypted hash to the original certificate. Then other sites that want to check on the validity of your certificate can use the Certificate Authority's well-published certificate (containing the Certificate Authority's public key) to verify the hash and thus verify that the contents of your supposed certificate match the contents that were seen and signed off on by the Certificate Authority. Assuming that the other site trusts the Certificate Authority, this then gives them confidence in your certificate.</p> <p>See Also Certificate, Certificate Authority, TLS (Transport Layer Security).</p>
Slave channel program	<p>Any program which dequeues messages from the MTA's message queues.</p> <p>See Also Master channel program.</p>
SMTP (Simple Mail Transfer Protocol)	<p>See RFC 821, or its update, RFC 5321.</p>
Spam	<p>Unsolicited bulk messages, especially e-mail messages, and usually undesired. The more formal term is Unsolicited Bulk E-mail (UBE). The term "spam" is believed to have originated from a Monty Python Flying Circus sketch.</p>
SSL (Security Sockets Layer)	<p>This protocol was developed by Netscape and has been superseded by TLS, which is backward compatible with SSL.</p>

See Also [TLS \(Transport Layer Security\)](#).

Subaddress A subaddress consists of extra detail information in the [RFC 5322](#) "local-part" of an address (the portion to the left of the "@" sign"); the subaddress is typically encoded into the local-part by using a [separator character such as the plus character, +](#), and is subject to site-specific interpretation. Use of subaddresses can be a convenient way, to *e.g.*:

- Request [delivery directed to a named folder](#).
- Indicate that a message is being received due to [membership of some mailing list](#).
- Request other special delivery handling, such as delivery to a voice mailbox.

See Also [local-part](#).

Symmetric encryption When encryption is done such that the same key encrypts and decrypts the data, it is said that the encryption is symmetric. This does require that the key that is used to encrypt the data must be given to the decryption side in a secure fashion.

T

Tar-pitting Intentionally slowing down SMTP responses: every response dragged out slowly, as if in a tar pit.

TLD (Top Level Domain) The permissible last part of domain names. For MTA purposes, see the [tlds.txt file](#).

TLS (Transport Layer Security) See [RFC 2246 \(The TLS Protocol\)](#).

Traffic analysis Analysis of who is sending what to whom; in an e-mail context, who (what senders) are sending what e-mail messages (how big, etc.), to what recipients. In some contexts, such information may be considered useful (or sensitive) even in the absence of specific information about exact message contents: for instance, merely knowing whether communications have increased or decreased between two correspondents provide useful competitive information. Monitoring, especially [message logging](#), may be useful in obtaining traffic information on an MTA system. Use of [BSMTP channels](#) may, in some cases, be of interest in blocking others from performing traffic analysis on your own messages.

U

UA (User Agent) User agent; *e.g.*, the Messenger Express e-mail client, or the Convergence e-mail client.

UBE (Unsolicited Bulk E-mail) Unsolicited bulk messages, especially e-mail messages, and usually undesired.
See Also [Spam](#).

V

Vanity domain	An apparent domain for which addresses are supported, but which does not have a domain entry in the LDAP directory (nor is it a domain alias); instead it is supported more as a sort of analogue to an alias on a user entry (typically by placing an attribute <code>msgVanityDomain</code> on the user entry). The use of vanity domains is strongly discouraged; proper domain entries (or domain aliases and analogues such as domain "uplevel" support) is preferred.
VERP (Variable Envelope Return Path)	Variable Envelope Return Path (VERP) refers to emitting messages with each recipient's version of the message copy having a different variant of the envelope From (envelope return path). One use is for mailing list messages, so that a list maintainer can correlate bounces of messages to the list back to which list member had a delivery problem.
Virtual domain	When a system hosts multiple domain names, it is considered to be supporting virtual domains---pseudodomain names that do not correspond to a system dedicated to only that domain name. Such setups are routine for modern Messaging Server deployments, and are typically provisioned via domain entries in an LDAP directory.



Index

Symbols

- "I" channel, See Local channel, 52–1
- \$SERVERROOT
 - Used to construct default dbtmpdir location, 16–11
 - Used to construct default lockdir location, 7–10
- `\${text}` substitution in mapping table, 37–16
- % routing, 34–4
- &/IMTA_DEFAULTDOMAIN/ substitution
 - Local channel rewrite rule, 34–15
- &/IMTA_HOST/ substitution
 - Conversion channel rewrite rule, 38–6
 - ldap_local_host MTA option, 39–84, 39–98, 51–3
 - Local channel rewrite rules, 34–15
- .HELD files
 - \$H flag in address access mapping tables, 44–9
 - \$V flag in FORWARD mapping table, 35–60
 - delivery_option clause, 39–93
 - Diagnosing, 52–11
 - directoryscan SNMP option, 60–2
 - HeldCount MTA counter, 55–23
 - Hold channel, 52–10
 - Hold channel and delivery_option clause, 39–93
 - holdlimit channel option, 33–58, 33–89, 33–103, 33–113
 - mailDomainStatus of hold, 7–7, 39–145
 - mailUserStatus of hold, 39–115
 - max_mime_levels MTA option, 39–209
 - max_mime_parts MTA option, 39–209
 - Not tracked by Job Controller, 42–3
 - QUARANTINE_ACTION Milter option, 45–5
 - Releasing from hold channel, 52–11
 - Sieve hold action, 5–18, 5–51
 - syslog notice
 - held_sndopr MTA option, 39–220, 39–246

A

- acceptalladdresses channel option, 33–28
- acceptvalidaddresses channel option, 33–28
- Access control, 44–1
 - Access mapping tables, 44–2
 - ACIs on LDAP attributes, 39–113
 - AUTH_REWRITE mapping table, 44–3
 - BURL_ACCESS mapping, 49–7
 - Client access to POP, IMAP, and MSHTTP servers, 20–1
 - Denial of service attacks
 - Defending against, 44–17
 - FROM_ACCESS mapping table, 44–2

Mailing lists

- Interpretation of multiple controls, 36–2
- MAIL_ACCESS mapping table, 44–2
- ORIG_MAIL_ACCESS mapping table, 44–2
- ORIG_SEND_ACCESS mapping table, 44–2
- PORT_ACCESS mapping table, 44–2, 44–3
- Recipient access mapping tables, 44–7
- Recipient restrictions, 44–2
- Sender restrictions, 44–2
- SEND_ACCESS mapping table, 44–2
- SMTP connections, 44–1
- TCP wrappers, 6–1
 - Filter syntax, 6–2
- Access mapping tables, 44–2
 - Interaction and timing, 44–15
 - MTA options, 39–188
- access_auth MTA option, 39–188
- access_counts MTA option, 39–188
 - *_ACCESS mapping table probes, 44–7
- access_errors MTA option, 39–156
 - Recipient *_ACCESS mapping table rejections, 44–8
 - Spam/virus filter package recipient rejections, 39–157
- access_orcpt MTA option, 39–189
 - *_ACCESS mapping table probes, 44–7
 - SRS and relay blocking, 49–50
- Accounts
 - Dispatcher worker processes run under
 - DEPRECATED: user Dispatcher option, 41–10
 - DEPRECATED: user Dispatcher service option, 41–10
 - Message Store
 - serveruid Base option, Replaced by user option in restricted.cnf, 7–11
 - Message Store admin
 - admins Message Store option, 16–9
 - admins Message Store option, MSHTTP support for SMTP AUTH, 33–150
 - httpproxyadmin MSHTTP option, 29–7
 - httpproxyadmin MSHTTP option, DEPRECATED, 29–7
 - imapadmin Proxy option, 26–1
 - proxyadmin base option, 7–15
 - smtpauthuser MSHTTP option, 29–6
 - storeadmin MMP/IMAP Proxy/POP Proxy/vdomain option, 27–27
 - Message Store indexer administrators
 - indexeradmins Message Store option, 16–9
 - Message Store public shared folders owner
 - publicsharedfolders Message Store option, 16–25
 - Message Store service administrator group

- serviceadmingroupdn Message Store option, 16–13
- MTA user
 - imsimta crdb utility, 58–19
 - imta_user MTA Tailor option, Replaced by user option in restricted.cnf, 40–10
 - Pipe channel command processing, 52–16
 - uid, Defragment database, 52–6
 - uid, Vacation response files, 39–67
- Pipe channel runs under
 - pipeuser option in restricted.cnf, 33–62, 33–107
 - user channel option, 33–62, 33–107
- User/group administrator
 - ugldapbinddn base option, 7–14
- accounturl base option, 7–8
- ackpolicy Message Store dbreplicate option, 16–17
- acktimeout Message Store dbreplicate option, 16–17
- actionattributes IMAP/POP/MessageTrace option, 21–11, 23–1
- actions IMAP/POP/MessageTrace option, 21–11, 23–1
- activate messageTrace option, 23–1, 33–86
 - ims-ms channel debugging, 51–6, 51–7
- active PAB option, 59–1
- additional_host_names channel option, 33–80
- addlineaddr channel option, 33–29
- Address
 - Intermediate
 - Reported in "capture :journal" 2003 format message copies, 54–14
 - Reported in "capture :journal" 2007 format message copies, 54–16
- Address access mapping tables
 - Conversion tags, 44–9
 - Enabling channel debugging, 44–9
 - Flags
 - List of, 44–9
 - FROM_ACCESS, 44–13
 - Header addition, 44–9
 - MAIL_ACCESS, ORIG_MAIL_ACCESS, SEND_ACCESS, and ORIG_SEND_ACCESS, 44–7
 - Message capture, 44–9
 - Message recipient limits, 44–9
 - Message size limits, 44–9
 - Sieve filter effects, 44–9
 - SMTP protocol delays, 44–9
 - Spam level setting, 44–9
 - syslog notice generation, 44–9
 - Testing of, 37–23
- Address reversal
 - Channel switch effect, 35–50
 - Conversion tags, 35–50
 - Direct LDAP lookups
 - Caching, 39–153
 - Performance tuning, 39–153
 - LDAP attribute triggered message capture, 54–6
 - LDAP lookups, 35–49
 - Message archiving, 35–50
 - Message capture, 35–50
 - Message size limits, 35–50
 - MTA options
 - reverse_envelope, 39–60
 - NOTARY flags, 35–50
 - Notification language preference, 35–50
 - Postmaster address
 - Per-domain, 35–50
 - REVERSE mapping table, 35–51
 - reverse_database_url MTA option, 35–52
 - reverse_envelope MTA option, 39–60
 - Spam/virus filter package optin, 35–50
 - use_reverse_database MTA option, 39–62, 39–200
- Addressbook (personal) lookups by the MTA, 39–182
- Addresses
 - % routing, 34–4
 - *-owner@*
 - Disables vacation message, 5–45, 44–15
 - *-request@*
 - Disables vacation message, 5–45, 44–15
- Authenticated sender
 - Adding to headers, 44–14
- Canonicalization, 35–48
 - reverse_url MTA option, 39–88
- Catchall
 - aliaswild channel option, 33–31, 35–45
 - alias_url1 MTA option, 39–86
 - ldap_domain_attr_catchall_address MTA option, 39–149
 - ldap_domain_attr_catchall_mapping MTA option, 39–149
 - mailDomainCatchallAddress LDAP attribute, 39–149
 - mailDomainCatchallMapping LDAP attribute, 39–149
- Characters
 - RFC 822 "specials" characters, In aliases, 35–25
- Comment strings
 - comment* channel options, 33–66
 - COMMENT_STRINGS mapping table, 35–55
 - mail_off MTA option, 39–185
 - post_off MTA option, 39–185

- Sieve filter address test :comment modifier, 5–21
- sourcecomment* channel options, 33–66
- Domain literals
 - Spaces in, 34–9
- Domain name omitted
 - Authentication, defaultdomain option, 27–12
 - Fixup of, 33–36, 33–67
 - log_local and logging of, 39–263
- EAI, G–4
- Eight bit characters in, 33–53
- Envelope From
 - Accepted,
 - error_text_accepted_return_address MTA option, 39–166
 - Authenticated sender as, 44–14
 - Blank, Distinguishing feature of notification messages, 47–1
 - Blank, returnenvelope channel option, 33–99
 - Domain corresponds to null MX, returnenvelope channel option, 33–99
 - Empty, Distinguishing feature of notification messages, 47–1
 - Invalid, error_text_invalid_return_address MTA option, 39–166
 - Mailing list override of, 35–32, 39–137
 - Mailing list postings and alias_envelope_from alias option, 35–14
 - Null, Distinguishing feature of notification messages, 47–1
 - Overridden via AUTH_ACCESS mapping, 49–38
 - Reply-to: addition, 35–37
 - setenvelopefrom Sieve action, 5–8, 5–61
 - SMS source, from_domain gateway_profile option, 53–5
 - Unknown,
 - error_text_unknown_return_address MTA option, 39–166
 - userswitchchannel effect, 33–82
 - Verifying apparently local addresses are valid, returnenvelope channel option, 33–99
 - Verifying apparently local addresses are valid, return_envelope MTA option, 39–156, 39–216
 - Verifying it rewrites to an MTA channel, returnenvelope channel option, 33–99
 - Verifying it rewrites to an MTA channel, return_envelope MTA option, 39–156, 39–215
 - Verifying its domain resolves in the DNS, returnenvelope channel option, 33–99
 - Verifying its domain resolves in the DNS, return_envelope MTA option, 39–156, 39–215
 - Envelope To
 - Domain aliases, 35–58
 - Fixup when domain name omitted, 33–36, 33–67
 - Processing of, 35–57
 - Rewrite rules, 35–58
 - Sieve filter access to, 36–6
 - Equal sign
 - token_char MTA option, 39–60, 39–246
 - Explicit routing
 - Interpretation of % and ! characters, 33–34
 - Removed with routelocal channel option, 33–42
 - Intermediate form
 - spamfilterN_final MTA options, 39–238
 - useintermediate channel option, 33–95
 - Internationalization, G–4
 - Length limit, 35–2, 35–24
 - LISTSERVE@*
 - Disables vacation message, 5–45, 44–15
 - MAILER-DAEMON@*
 - Disables vacation message, 5–45, 44–15
 - majordomo@*
 - Disables vacation message, 5–45, 44–15
 - MTA options, 39–54
 - owner-.*@*
 - Disables vacation message, 5–45, 44–15
 - Parsing
 - ap_debug MTA option, 39–73
 - Debugging, ap_debug MTA option, 39–73
 - Sieve filter address test, 5–21
 - Percent hack
 - Special rewriting of, 34–11
 - Personal name (RFC 822 "phrase")
 - *personal* channel options, 33–41, 33–76
 - ldap_personal_name MTA option, 39–120
 - Length limit, 35–3
 - MIME encoding and the charset8 channel option, 33–53
 - Not included in reverse database probes, 35–51
 - PERSONAL_NAMES mapping table, 35–55
 - Sieve filter address test :display modifier, 5–21
 - Postmaster, 47–25
 - \$H flag in REVERSE mapping table, 35–53
 - MDNs, RETURN_PERSONAL option in disposition_option.opt, 47–21
 - Per-domain, Address reversal, 35–50
 - user_case MTA option, 39–64
 - Quotation marks embedded in local-part, 34–19
 - Reversal, 35–48
 - Side effects (intended), 35–49

- RFC 1137 restricted encoding of, 33–40
 - restricted switch of test -rewrite utility, 58–37
- RFC 822 "specials" characters
 - In aliases, 35–25
- RFC 822 phrase, 35–55
- Source routes, 33–35
 - *exproute and *improute channel options, 33–38
 - delivery_option clauses, 39–94
 - Interpretation of % and ! characters, 33–34
 - Removing during rewriting, 34–8
 - Stripping, 33–37
- SRS encoding
 - Bad hash, error_text_srs_badhash MTA option, 39–163
 - Channel options, 33–29
 - MTA options, 39–244
 - Syntax errors, error_text_srs_syntax MTA option, 39–162
 - Time out, error_text_srs_timeout MTA option, 39–163
- Subaddresses
 - Address reversal, 35–54
 - Alias file, 35–24
 - ldap_domain_attr_subaddress MTA option, 39–144
 - Mailing list members, 36–22
 - Mailing list VERP type functionality, 39–138
 - Meta-groups, 39–100
 - Rewrite rule substitution, 34–20
 - Sieve filter subaddress extension, 5–43
 - subaddress* channel options, 33–42
- Syntax check not performed after rewriting, 34–8
- Testing of
 - test -rewrite utility, 58–28
- token_char MTA option, 39–60, 39–246
- Trailing dot on host/domain, 34–5
- user_case MTA option, 39–64
- UUCP style
 - Special rewriting of, 34–12
- Vacation messages
 - ldap_autoreply_addresses MTA option, 39–128
- addresssrs channel option, 33–29
- addrreturnpath channel option, 33–63
- addrspfile channel option, 33–57
- addrspjob channel option, 33–99
- addrtypescan channel option, 33–29, 33–108
 - "capture :journal" message copies, 54–16
 - HEADER_CHECK alias file named parameter, 35–35
 - ldap_check_header MTA option, 39–141
 - addrtypescanbccdefault channel option, 33–29, 33–108
 - ldap_check_header MTA option, 39–141
 - adminbypassquota IMAP option, 21–3
 - admins Message Store option, 16–9
 - affinitylist channel option, 33–137
 - after channel option, 33–100
 - BSMTP channels, 50–3
 - Alarm options, 11–1
 - noticehost, 11–1
 - noticeport, 11–1
 - noticercpt, 11–1
 - noticesender, 11–1
 - Postmaster address, 47–2
 - system group, 11–2
 - system:diskavail
 - description, 11–2
 - statinterval, 11–2
 - threshold, 11–3
 - thresholddirection, 11–3
 - warninginterval, 11–3
 - system:serverresponse
 - description, 11–2
 - statinterval, 11–2
 - threshold, 11–3
 - thresholddirection, 11–3
 - warninginterval, 11–3
- Alarms, 1
- Alias and address MTA options, 39–54
- Alias database, 35–42
 - alias_database_url MTA option, 39–202
 - Case insensitive, 35–42
 - Example, 35–42
 - Format of, 35–43
 - MTA options
 - alias_database_url MTA option, 39–202
 - comment_chars, 35–44
 - use_alias_database, 39–60
 - Used in addition to (not replacing) alias options or alias file, 35–42
 - World readable, 35–44
- Alias file
 - Maximum number of entries
 - alias_hash_size MTA option, 39–176
 - Named parameter, 35–26
- alias group, 35–8
- Alias options, 35–9
 - alias_alternate_recipient, 35–10
 - alias_and, 35–10
 - alias_auth_channel, 35–10
 - alias_auth_list, 35–10
 - alias_auth_mapping, 35–10
 - alias_auth_username, 35–10

- alias_blocklimit, 33–112, 35–11
 - error_text_list_block_over MTA option, 39–159
 - error_text_user_block_over MTA option, 39–160
- alias_cant_channel, 35–10
- alias_cant_list, 35–10
- alias_cant_mapping, 35–10
- alias_cant_username, 35–10
- alias_capture, 35–11
- alias_conversion_tag, 35–11
- alias_creation_date, 35–12
- alias_deferred, 35–12
- alias_deferred_list, 35–12
- alias_deferred_mapping, 35–12
- alias_delay_notifications, 35–13
- alias_digest_recurrence, 35–13
- alias_direct_list, 35–14
- alias_direct_mapping, 35–14
- alias_entry, 35–9
- alias_envelope_from, 35–14
- alias_error_text, 35–14
- alias_expandable, 35–15
- alias_expiry, 35–15
- alias_filter, 35–15
 - Sieve hierarchy, 5–65
- alias_header_addition, 35–15
 - Compared to use of mgrpAddHeader group LDAP attribute, 39–139
- alias_header_alias, 35–16
- alias_header_check, 35–16
- alias_header_expansion, 35–16
- alias_header_trim, 35–15
 - Compared to use of mgrpRemoveHeader group LDAP attribute, 39–139
- alias_hold_list, 35–16
- alias_hold_mapping, 35–16
- alias_importance, 35–17
- alias_journal, 35–11
- alias_keep_delivery, 35–17
- alias_keep_read, 35–17
- alias_linelimit, 33–112, 35–11
 - error_text_list_line_over MTA option, 39–160
 - error_text_user_line_over MTA option, 39–160
- alias_list_name, 35–17
- alias_nodelay_notifications, 35–13
- alias_nohold_list, 35–16
- alias_nohold_mapping, 35–16
- alias_nonexpandable, 35–15
 - expn* channel options, 33–126
- alias_nooriginator_reply, 35–18
- alias_noreceivedfor, 35–19
- alias_noreceivedfrom, 35–19
- alias_nosolicit, 35–19
- alias_optin1, 35–19
- alias_or, 35–10
- alias_originator_reply, 35–18
- alias_password, 35–20
- alias_precedence, 35–17
- alias_prefix_text, 35–20
- alias_priority, 35–17
- alias_private, 35–20
- alias_public, 35–20
- alias_receivedfor, 35–19
- alias_receivedfrom, 35–19
- alias_reprocess, 35–21
- alias_sasl_auth_list, 35–21
- alias_sasl_auth_mapping, 35–22
- alias_sasl_cant_list, 35–22
- alias_sasl_cant_mapping, 35–22
- alias_sasl_mdoerator_list, 35–22
- alias_sasl_moderator_mapping, 35–22
- alias_sensitivity, 35–17
- alias_sequence_prefix, 35–22
- alias_sequence_strip, 35–22
- alias_sequence_suffix, 35–22
- alias_single, 35–22
- alias_spare*, 35–22
- alias_suffix_text, 35–20
- alias_tag, 35–23
- alias_to, 35–23
- alias_username, 35–23
- alias_username_auth_list, 35–10
- alias_username_cant_list, 35–10
- Header addition modifiers, 35–46
- Values
 - URL types, 1–4

- aliasdetourhost channel option, 33–30, 33–60
 - Alternate conversion channel, 38–5
 - Routing to a gateway system, 49–47
- aliasdetourhost_null_optin MTA option, 39–91
- Aliases, 35–2
 - Alias file, 35–23
 - Backslash character, 35–24
 - Colon character, 35–24
 - Comment line, 35–25
 - Compiling, 35–26
 - Continuation lines, 35–24
 - Duplicate left hand sides not allowed, 35–25
 - Format, 35–24
 - Including additional files, 35–26
 - LDAP URL alias values, 35–41
 - Mailing lists, 35–40
 - Named parameters, 35–24
 - Protection of include files, 35–26

- Restrictions, 35–47
 - Subaddresses, 35–24
- alias group, 35–8
- alias_case MTA option, 39–55
- alias_domains MTA option, 39–56
- alias_magic MTA option, 39–56
- Case sensitivity
 - alias_case MTA option, 39–55
 - Insensitive matching in alias database, 35–42
- Creation date
 - alias_creation_date alias option, 35–12
 - CREATION_DATE alias file named parameter, 35–31
- Database
 - Format of probes, alias_domains MTA option, 39–56
- Disabled alias
 - error_text_disabled_alias MTA option, 39–161
- Duplicates (left hand side) not allowed in alias file, 35–25
- Duplicates in LDAP
 - error_text_duplicate_addr MTA option, 39–162
- File
 - Format of probes, alias_domains MTA option, 39–56
- Header addition modifiers, 35–46
- LDAP, 35–5
 - alias_urlN MTA options, 39–85
 - Overview, 35–3
 - Special formats, 35–45
- ldap_alias_addresses MTA option, 39–121
- ldap_equivalence_addresses MTA option, 39–121
- MTA options, 39–54
 - string_pool_size_2, 39–181
- Named parameters, 35–26
 - Header addition modifiers, 35–46
- No valid translation values
 - error_text_empty_alias MTA option, 39–162
- Options
 - See Alias options, 35–9
- Postmaster, 35–9
- Recursive definition, 35–46
 - max_alias_levels MTA option, 35–46, 39–58
- Recursive definition of, 35–42
- Restrictions, 35–47
- string_pool_size_2 MTA option, 39–181
- Subaddresses
 - subaddress* channel options, 33–42
- Subaddresses in, 35–44, 35–45
- Testing if found in *_ACCESS mapping table entries, 44–13
- Testing of
 - test -rewrite utility, 58–28
- Unified Configuration, 35–8
- Used in notification messages (DSNs and MDNs), 35–24
- aliaslocal channel option, 33–31, 35–45, 39–56
 - Compared to local channel, 52–1
 - Compared to localbehavior, 33–39
- aliasmagic channel option, 33–31
 - Aliases in LDAP, 35–5
 - Override via \$nT rewrite rule control sequence, 34–32
- aliasoptindetourhost channel option, 33–30, 33–60
 - aliasdetourhost_null_optin to selectively disable effect, 39–91
- aliaspostmaster channel option, 33–93
- aliaswild channel option, 33–31, 39–56
- alias_alternate_recipient alias option, 35–10
- alias_and alias option, 35–10
- alias_auth_channel alias option, 35–10
- alias_auth_list alias option, 35–10
- alias_auth_mapping alias option, 35–10
- alias_auth_username alias option, 35–10
- alias_blocklimit alias option, 33–112, 35–11
 - error_text_list_block_over MTA option, 39–159
 - error_text_user_block_over MTA option, 39–160
- alias_cant_channel alias option, 35–10
- alias_cant_list alias option, 35–10
- alias_cant_mapping alias option, 35–10
- alias_cant_username alias option, 35–10
- alias_capture alias option, 35–11
- alias_case MTA option, 39–55
- alias_conversion_tag alias option, 35–11
- alias_creation_date alias option, 35–12
- alias_database_url MTA option, 39–202
- alias_deferred alias option, 35–12
- alias_deferred_list alias option, 35–12
- alias_deferred_mapping alias option, 35–12
- alias_delay_notifications alias option, 35–13
- alias_digest_recurrence alias option, 35–13
- alias_direct_list alias option, 35–14
- alias_direct_mapping alias option, 35–14
- alias_domains MTA option, 39–56
 - Compared with aliaswild channel option, 33–31
- alias_entry alias option, 35–9
- alias_entry_cache_negative MTA option, 39–152
- alias_entry_cache_size MTA option, 39–152
- alias_entry_cache_timeout MTA option, 39–152
 - Lag in seeing LDAP alias changes take effect, 35–48
- alias_envelope_from alias option, 35–14
- alias_error_text alias option, 35–14
- alias_expandable alias option, 35–15

alias_expiry alias option, 35–15

alias_filter alias option, 35–15

- Sieve hierarchy, 5–65

alias_hash_size MTA option, 39–176

alias_header_addition alias option, 35–15

alias_header_alias alias option, 35–16

alias_header_check alias option, 35–16

alias_header_expansion alias option, 35–16

alias_header_trim alias option, 35–15

alias_hold_list alias option, 35–16

alias_hold_mapping alias option, 35–16

alias_importance alias option, 35–17

alias_journal alias option, 35–11

alias_keep_delivery alias option, 35–17

alias_keep_read alias option, 35–17

alias_linelimit alias option, 33–112, 35–11

- error_text_list_line_over MTA option, 39–160
- error_text_user_line_over MTA option, 39–160

alias_list_name, 35–17

alias_magic MTA option, 39–56

- Aliases in LDAP, 35–5
- Override via \$nT rewrite rule control sequence, 34–32

alias_member_size MTA option, 39–176

alias_nodelay_notifications alias option, 35–13

alias_nohold_list alias option, 35–16

alias_nohold_mapping alias option, 35–16

alias_nonexpandable alias option, 35–15

- expn* channel options, 33–126

alias_nooriginator_reply alias option, 35–18

alias_noreceivedfor alias option, 35–19

alias_noreceivedfrom alias option, 35–19

alias_nosolicit alias option, 35–19

alias_optin1 alias option, 35–19

alias_or alias option, 35–10

alias_originator_reply alias option, 35–18

alias_password alias option, 35–20

alias_precedence alias option, 35–17

alias_prefix_text alias option, 35–20

alias_priority alias option, 35–17

alias_private alias option, 35–20

alias_public alias option, 35–20

alias_receivedfor alias option, 35–19

alias_receivedfrom alias option, 35–19

alias_reprocess alias option, 35–21

alias_sasl_auth_list alias option, 35–22

alias_sasl_auth_mapping alias option, 35–22

alias_sasl_cant_list alias option, 35–22

alias_sasl_cant_mapping alias option, 35–22

alias_sasl_moderator_list alias option, 35–22

alias_sasl_moderator_mapping alias option, 35–22

alias_sensitivity alias option, 35–17

alias_sequence_prefix alias option, 35–22

alias_sequence_strip alias option, 35–22

alias_sequence_suffix alias option, 35–22

alias_single alias option, 35–22

alias_spare* alias options, 35–22

alias_suffix_text alias option, 35–20

alias_tag alias option, 35–23

alias_to alias option, 35–23

alias_url0 MTA option

- Example, 35–7

alias_urlN MTA options, 39–85

- Aliases in LDAP, 35–5
- Alternative to alias file LDAP URL alias values, 35–41

alias_username alias option, 35–23

alias_username_auth_list alias option, 35–10

alias_username_cant_list alias option, 35–10

allowanonymouslogin IMAP option, 21–10

allowanonymouslogin MSHTTP option, 29–11

allowanonymouslogin POP option, 22–2

allowcollect MSHTTP option, 29–5

allowetrn channel option, 33–116

allowldapaddresssearch MSHTTP option, 29–4

allowswitchchannel channel option, 33–81

- Alternate conversion channel, 38–4

allow_unquoted_addrs_violate_rfc2798, 39–92

alternateblocklimit channel option, 33–86, 33–111

alternatetchannel channel option, 33–86, 33–111

alternatelinelimit channel option, 33–86, 33–111

alternaterecipientlimit channel option, 33–86, 33–111

altservice MSHTTP option, 29–11

alwaysencrypt smime option, 30–3

always sign smime option, 30–4

alwaysusedefaulthost PAB option, 59–1

annotatemail notifytarget option, 24–6

APOP

- crams mmp/imapproxy/popproxy/vdomain option, 27–11
- has_plain_passwords auth option, 12–1
- userPassword LDAP attribute
- Contain clear-text password, 39–103

appletlogging smime option, 30–7

Application information

- Syntax of, 55–8

APPLICATIONINFO environment variable

- test_smtp_master and test_smtp_slave use of, 52–9

ap_debug MTA option, 39–73

Archive package integration

- DEBUG Archive option, 45–9
- DESTINATION Archive option, 39–92, 39–117, 45–9
- DIRECTORY Archive option, 45–9

- IDSUFFIX Archive option, 45–9
- MODE Archive option, 45–9
- POSTEDDATEMODE Archive option, 45–9
- RESETDEBUG Archive option, 45–9
- REVERSE Archive option, 45–9
- SOURCE_CHANNEL Archive option, 45–9
- spamfilterN_config_file options, 45–9
- STYLE Archive option, 45–9
- SUBDIRS Archive option, 45–9
- TRUSTEXISTINGHASH Archive option, 45–9
- USEHEADERRECIPIENTS Archive option, 45–9
- Archiving, 54–17
 - AXS:One integration, 54–20
 - Architecture, 54–20
 - MTA configuration, 54–21
 - MTA configuration, Example, 54–22
 - MTA support, 54–21
 - Compliance, 54–17
 - IMAP APPEND operations, 16–13
 - Message Store archive options, 16–13
 - Journal format, 16–14
 - MESSAGE-SAVE-COPY mapping table, 54–3
 - MTA configuration
 - AXS:One integration, 54–21
 - Message identifier generation, 54–18
 - Operational, 54–17
 - See Archive package integration, 39–203
 - See Message, Archiving, 39–203
 - Which messages to archive, 54–17
- Archiving
 - Message identifier generation, 54–18
- async MeterMaid option, 46–4
- Attachments
 - gzip
 - gzipattach MSHTTP option, 29–6, 29–6, 29–6
 - safe-tcl
 - safe_tcl_mode MTA option, 39–273
 - See also Character set, Conversion, 38–16
 - See also Message, Conversions, 38–1
 - See also MIME, 39–273
- attributelist PAB option, 59–1
- Auth options, 1, 12–1
 - authenticationldapattributes, 27–6
 - authenticationserver, 27–6
 - auto_transition, 12–1
 - broken_client_login_charset, 12–2
 - canonicalsearchfilter, 12–1
 - has_plain_passwords, 12–1
 - requireauthenticationserver, 27–21
 - searchfilter, 12–1
 - searchfordomain, 12–2
 - usedomainmap, 12–1
- authcachesize base option, 7–3
- authcachettl base option, 7–3, 27–6
- authcachettl MMP/IMAP Proxy/POP Proxy/vdomain option, 27–5
- Authentication
 - Auth options, 12–1
 - Caching
 - authcachesize base option, 7–3
 - authcachettl base option, 7–3, 27–6
 - defaultdomain option, 27–12
 - Mechanism
 - EXTERNAL, implicitaslexternal channel option, 33–151
 - PORT_ACCESS mapping table, 44–2
 - Message Store
 - SASL library code used, 35–5, 39–103
 - SMTP AUTH
 - Channel options, 33–149
 - FROM_ACCESS mapping table, 44–2
 - implicitaslexternal channel option, 33–151
 - SASL library code used, 35–5, 39–103
- authenticationldapattributes auth option, 27–6
- authenticationldapattributes IMAP Proxy/POP Proxy option, 27–6
- authenticationldapattributes vdomain option, 27–6
- authenticationserver auth option, 27–6
- authenticationserver IMAP Proxy/POP Proxy option, 27–6
- authpassword channel option, 33–146
- authrewrite channel option, 33–32, 33–63, 33–147, 33–147
 - AUTH_REWRITE mapping table, 33–32, 33–64, 33–147
 - FROM_ACCESS mapping table, 44–14
- authservice POP proxy/Virtual Domain option, 27–6
- authservicettl POP proxy/Virtual Domain option, 27–6
- authusername channel option, 33–146
- autodetect Message Store deadlock option, 16–17
- autorepair Message Store option, 16–9
- autorepairdebug Message Store option, 16–10
- autoreply_timeout_default MTA option, 39–65
- Autorestart options, 7–22
 - enable, 7–23
 - timeout, 7–23
- auto_transition auth option, 12–1
- AXS:One
 - See Archive package integration, 45–9

B

- backlog Dispatcher option, 41–3
- backlog MeterMaid option, 46–4
- backlog SMS smpp_relay option, 53–9
- backlog SMS smpp_server option, 53–10

- backlog tcp_listen option, 28–1
- backoff channel option, 33–101
 - Defragmentation channel, 52–4
 - ims-ms channels, 51–1
 - Automatic override, 49–29
 - LMTTP client channels
 - Override via
 - MAILBOX_BUSY_FAST_RETRY, 49–29
- backsideport IMAP Proxy and POP Proxy option, 27–6
- Backslash
 - Continuation line indicator
 - In aliases file, 35–24
- Backup
 - backupexclude Message Store option, 16–9
- backupdir Message Store option, 16–4
- backupexclude Message Store option, 16–9
- backup_group options, 19–1, 19–1
- Bad guy penalty
 - bgdecay option, 7–4, 27–8
 - bgexcluded option, 7–4, 27–8
 - bglinear option, 7–4, 27–8
 - bgmax option, 7–4, 27–7
 - bgmaxbadness option, 7–4, 27–8
 - bgpenalty option, 7–4, 27–8
- bangonly channel option, 33–34
- bangoverpercent channel option, 33–34
 - Rewrite rule address interpretation, 34–3, 34–4
- bangstyle channel option, 33–35
- Banner
 - IMAP
 - banner IMAP option, 21–10
 - banner IMAP Proxy option, 27–7
 - POP
 - banner POP option, 22–2
 - banner POP Proxy option, 27–7
 - SMTP server
 - BANNER_PURGE_DELAY TCP/IP-channel-specific option, 49–21
 - CUSTOM_VERSION_STRING TCP/IP-channel-specific option, 49–23
 - Hostname used, 33–80
 - SMTP server (remote)
 - 5xx or 4xx error, X connection transaction log entries, 33–135
 - fire away, 33–118
 - MurkWorks, 33–118
 - Timeout awaiting, lastresort host not attempted, 33–62, 33–140
 - Whether EHLO is supported, *ehlo channel options, 33–118
 - SMTP SUBMIT
 - banner SUBMIT Proxy option, 27–7
 - SMTP/LMTP server
 - BANNER_ADDITION TCP/IP-channel-specific option, 49–21
 - BANNER_HOST TCP/IP-channel-specific option, 49–21
 - BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 - SMTP/LMTP server (remote)
 - Logging of, LOG_BANNER TCP/IP-channel-specific option, 49–27
 - Timeout awaiting,
 - STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
- banner IMAP option, 21–10
- banner IMAP Proxy option, 27–7
- banner MMP option, 27–7
- banner POP option, 22–2
- banner POP Proxy option, 27–7
- banner SUBMIT Proxy option, 27–7
- BANNER_HOST TCP/IP-channel-specific option
 - Local channel official_host_name, 52–2
- Base options, 1, 7–3
 - accounturl, 7–8
 - authcachsize, 7–3
 - authcachetl, 7–3, 27–5
 - autorestart group, 7–22
 - enable, 7–23, 7–23
 - bgdecay, 7–4, 27–8
 - bgexcluded, 7–4, 27–8
 - bglinear, 7–4, 27–8
 - bgmax, 7–4, 27–7
 - bgmaxbadness, 7–4, 27–8
 - bgpenalty, 7–4, 27–8
 - certmap group, 7–23
 - cmapldapattr, 7–24
 - dncomps, 7–23
 - filtercomps, 7–23
 - verifycert, 7–23
 - dblockcount, 7–10
 - dbtxnsync, 7–9
 - dcroot, 7–16
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
 - debugkeys, 7–4, 27–12
 - defaultdomain, 7–4, 27–13
 - Default for contextname SNMP option, 60–2
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
 - dnsresolveclient, 7–16
 - domainmap group, 7–3
 - debug, 7–3, 7–22
 - enablelastaccess, 7–9
 - filterurl, 7–8

folderurl, 7-8
 hostname, 7-9
 Default for http.smtphost option, 29-10
 Direct LDAP alias lookups, 35-6
 ldap_local_host MTA option, 39-84, 39-98
 installedlanguages, 7-8
 ipv6in, 7-17, 27-15
 ipv6out, 7-17, 27-15
 ipv6sortorder, 7-18
 ipv6usegethostbyname, 7-18
 ldapcheckcert, 7-9
 ldapconnecttimeout, 7-10
 Direct LDAP alias lookups, 35-5
 ldapmodifytimeout, 7-10
 ldappoolrefreshinterval, 7-10
 ldaprequiretls, 7-15
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 ldapsearchtimeout, 7-10
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 ldaptrace, 7-10
 ldap_basedn_filter_schema1, 7-7, 39-82, 39-89
 ldap_basedn_filter_schema2, 7-7, 39-82, 39-89
 ldap_domain_attr_alias, 7-6, 39-143
 ldap_domain_attr_basedn, 7-6, 39-143
 ldap_domain_attr_mail_status, 7-7, 39-145
 ldap_domain_attr_status, 7-7, 39-145
 ldap_domain_attr_uid_separator, 7-6, 39-144
 ldap_domain_filter_schema1, 7-8, 39-83, 39-89
 Direct LDAP domain lookups, 34-30
 ldap_domain_filter_schema2, 7-8, 39-83, 39-89
 Direct LDAP domain lookups, 34-30
 ldap_domain_known_attributes, 7-5, 39-83
 Direct LDAP domain lookups, 34-30
 ldap_domain_timeout, 7-5, 39-83, 39-153
 TCP wrappers, 6-2
 ldap_host_alias_list, 7-9
 ldap_schemalevel, 7-5, 39-90
 listenaddr, 7-17
 ENS server host, 61-1
 listurl, 7-8
 lockdir, 7-10
 loginseparator, 7-16
 obsoleteimap, 7-11
 preferpoll, 7-15, 27-20
 projectid, 7-18
 proxyadmin, 7-15
 Host-specific override by imapadmin option,
 26-1
 proxyadminpass, 7-15
 Host-specific override by imapadminpass
 option, 26-1
 proxyimapport, 7-16
 proxyimapssl, 7-16
 proxyserverlist, 7-16
 proxytrustmailhost, 7-16
 pwchangeurl, 7-8, 7-9
 rbac, 7-16
 rfc822headerallow8bit, 7-15
 secret, 7-15
 serveruid, 7-11
 softtokendir, 7-17
 ssladjustciphersuites, 7-11, 27-23
 sslcachedir, 7-3, 27-25
 sslcompress, 7-13
 ssldebug, 7-12
 ssldbpath, 7-12
 ssldbprefix, 7-13
 sslnicknames, 7-17, 27-26
 sslpkix, 7-13
 sslrenegotiate, 7-19
 sslrequiresafenegotiate, 7-13
 sslv3enable, 7-13
 stressfdwait, 7-5
 stressperiod, 7-5
 supportedlanguages, 7-13
 thresholddelay, 7-13
 tlsv12enable, 7-18
 tmpdir, 7-14
 ugldapbasedn, 7-14
 Direct LDAP alias lookups, 35-6
 ugldapbindcred, 7-14
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 ugldapbinddn, 7-14
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 ugldaphost, 7-14
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 Mapping table \$]ldap-url[substitutions,
 37-14
 ugldapport, 7-14
 Direct LDAP alias lookups, 35-5
 Direct LDAP domain lookups, 34-29
 Mapping table \$]ldap-url[substitutions,
 37-14
 ugldapusessl, 7-15
 Direct LDAP alias lookups, 35-5
 welcomemsg, 7-17
 bgdecay option, 7-4, 27-8
 bgexcluded option, 7-4, 27-8
 bglinear option, 7-4, 27-8
 bgmax option, 7-4, 27-7
 bgmaxbadness option, 7-4, 27-8

bgpenalty option, 7–4, 27–8
 bidirectional channel option, 33–102
 Binary attachments
 Macintosh files, 38–21
 binaryclient channel option, 33–117
 binaryserver channel option, 33–117
 Bitbucket channel, 52–2
 Configuration, 52–2
 discard or jettison actions, 5–23
 MTA message transaction log entries, 52–2
 Routing via address access mapping tables, 44–9
 blocked_mail_from_ips MTA option, 39–155
 blocketrn channel option, 33–116
 blocklimit
 Effect set via address access mapping tables, 44–9
 blocklimit channel option, 33–112, 39–159
 Notification messages, 47–25
 block_limit MTA option, 33–112, 39–205
 block_size MTA option, 39–206
 alias_blocklimit alias option, 35–11
 [BLOCKLIMIT] alias file named parameter, 35–30
 block_time local_table MeterMaid option, 46–2
 bodytextonly indexer option, 25–2
 Botnet attack
 Blocking via LOG_ACTION, 55–19
 bounce_block_limit MTA option, 39–206, 39–213
 Brightmail
 See Spam/virus filter package integration, Brightmail, 45–2
 broken_client_login_charset auth options, 12–2
 BSMTP channels, 50–1
 BSIN channels, 50–1
 BSOUT channels, 50–1
 Configuration, 50–1
 Service conversions, 50–4
 BSMTP-specific channel options, 33–51
 Buffer overruns
 Content-disposition: header lines, 33–49
 Content-type: header lines, 33–49
 buffer_size MTA option, 39–171
 BURL
 U modifier in MTA message transaction log entries, 55–4

C
 cacheeverything channel option, 33–135
 cachefailures channel option, 33–135
 cachepath partition option, 18–1, 18–1
 cachepreviewlen Message Store option, 16–12
 cachesuccesses channel option, 33–135
 cachesynclevel Message Store option, 16–5
 cachettl SNMP option, 60–2
 cache_debug MTA option, 39–73
 cache_magic MTA option -- OBSOLETE, 39–171
 Calendar invitations
 msexchange channel option, 33–49, 33–130, 33–152
 canonicalsearchfilter auth option, 12–1
 canonicalvirtualdomaindelim MMP/IMAP Proxy/POP Proxy option, 27–9
 capability_acl IMAP option, 21–5
 capability_annotate IMAP option, 21–5
 capability_binary IMAP option, 21–5
 capability_catenate IMAP option, 21–5
 capability_children IMAP option, 21–5
 capability_condstore IMAP option, 21–5
 capability_context_search IMAP option, 21–5
 capability_context_sort IMAP option, 21–5
 capability_create_special_use IMAP option, 21–8
 capability_enable IMAP option, 21–6
 capability_esearch IMAP option, 21–6
 capability_esort IMAP option, 21–6
 capability_id IMAP option, 21–6
 capability_idle IMAP option, 21–6
 capability_imap4 IMAP option, 21–6
 capability_imap4rev1 IMAP option, 21–6
 capability_language IMAP option, 21–6
 capability_list_status IMAP option, 21–6
 capability_literal IMAP option, 21–6
 capability_login_referrals IMAP option, 21–7
 capability_metadata IMAP option, 21–7
 capability_multisearch IMAP option, 21–7
 capability_namespace IMAP option, 21–7
 capability_notify IMAP option, 21–7
 capability_qresync IMAP option, 21–7
 capability_quota IMAP option, 21–8
 capability_sasl_ir IMAP option, 21–8
 capability_searchres IMAP option, 21–8
 capability_sort IMAP option, 21–8
 capability_sort_display IMAP option, 21–8
 capability_special_use IMAP option, 21–8
 capability_starttls Deploymap option, 14–2
 capability_starttls IMAP option, 21–8
 capability_thread_references IMAP option, 21–9
 capability_thread_subject IMAP option, 21–9
 capability_uidplus IMAP option, 21–9
 capability_unselect IMAP option, 21–9
 capability_urlauth IMAP option, 21–9
 capability_within IMAP option, 21–9
 capability_xrefresh IMAP option, 21–10
 capability_xsender IMAP option, 21–10
 capability_xserverinfo IMAP option, 21–10
 capability_xum1 IMAP option, 21–10
 capability_x_netscape IMAP option, 21–9

capability_x_orcl_as IMAP option, 21–9
 capability_x_sun_imap IMAP option, 21–9
 capability_x_sun_sort IMAP option, 21–10
 capability_x_unauthenticate IMAP option, 21–10
 caption channel option, 33–56
 capture_format_default MTA option, 39–92
 Case sensitivity
 Host/domain names not case sensitive per RFC 822, 34–5
 Certificate
 certmap options, 7–23
 certurl smime option, 30–2
 cert_enable MSHTTP option, 29–7
 Client authentication
 certmapfile option, DELETED, 27–9
 Debugging
 certmap debugkeys value, 27–12
 Nicknames
 CLIENT_CERT_NICKNAME TCP/IP-channel-specific option, 49–22
 sslnicknames base option, 7–17, 27–26
 sslnicknames ENS option, 27–27, 61–2
 sslnicknames MTA option, 27–27, 39–218
 sslnicknames option, 27–26
 sslnicknames POP option, 22–4, 27–27
 Revocation List
 cert_enable MSHTTP option, 29–7
 cert_port MSHTTP option, 29–7
 checkoverssl S/MIME option, 30–5
 crlaccessfail S/MIME option, 30–5
 crl_dir smime option, 30–4
 crlenable smime option, 30–4
 crlmappingurl smime option, 30–5
 crlurllogindn smime option, 30–4
 crlurlloginpw smime option, 30–5
 crlusepastnextupdate S/MIME option, 30–7
 readsigncert S/MIME option, 30–6
 revocationunknown S/MIME option, 30–6
 sendencryptcert S/MIME option, 30–6
 sendencryptcertrevoked S/MIME option, 30–6
 sendsigncert S/MIME option, 30–7
 sendsigncertrevoked S/MIME option, 30–7
 SMTP TLS
 Required for implicit_sasl_external to take effect, 33–151
 TLS_ACCESS mapping table
 MTA decline to permit TLS use, 49–45
 Validity
 IGNORE_BAD_CERT TCP/IP-channel-specific option, 49–27
 ldapcheckcert base option, 7–9
 Required for implicit_sasl_external to take effect, 33–151
 Certificate Authority, G–1
 Certificate server
 msprobe probe of, 10–2
 Certmap options
 cmapldapattr, 7–24
 dncomps, 7–23
 filtercomps, 7–23
 verifycert, 7–23
 certurl smime option, 30–2
 cert_enable MSHTTP option, 29–7
 changeflag notifytarget option, 24–6
 Channel block definitions
 Testing of
 test -rewrite utility, 58–28
 channel group, 33–5
 Channel options, 33–6
 acceptalladdresses, 33–28
 acceptvalidaddresses, 33–28
 additional_host_names, 33–80
 addlineaddr, 33–29
 Addresses, 33–28
 addresssr, 33–29
 addrreturnpath, 33–63
 addrspfile, 33–57
 addrspjob, 33–99
 addrtypescan, 33–29, 33–108
 "capture :journal" message copies, 54–16
 HEADER_CHECK alias file named parameter, 35–35
 ldap_check_header MTA option, 39–141
 addrtypescanbccdefault, 33–29, 33–108
 ldap_check_header MTA option, 39–141
 affinitylist, 33–137
 after, 33–100
 BSMTP channels, 50–3
 aliasdetourhost, 33–30, 33–60
 Alternate conversion channel, 38–5
 Routing to a gateway system, 49–47
 aliaslocal, 33–31, 35–45, 39–56
 Compared to local channel, 52–1
 Compared to localbehavior, 33–39
 aliasmagic, 33–31
 Aliases in LDAP, 35–5
 Override via \$nT rewrite rule control sequence, 34–32
 aliasoptindetourhost, 33–30, 33–60
 aliaspostmaster, 33–93
 aliaswild, 33–31, 39–56
 allowetrn, 33–116
 allowswitchchannel, 33–81
 Alternate conversion channel, 38–4
 Alphabetic list, 33–7
 alternatblocklimit, 33–86, 33–111

alternatechannel, 33–86, 33–111
 alternatelinelimit, 33–86, 33–111
 alternatereipientlimit, 33–86, 33–111
 Arguments
 Case preservation, 33–7
 Length limits, 33–7
 Quoting, 33–7
 URL types, 1–4
 Attachments and MIME processing, 33–45
 authpassword, 33–146
 authrewrite, 33–32, 33–63, 33–147
 AUTH_REWRITE mapping table, 33–32, 33–64, 33–147
 FROM_ACCESS mapping table, 44–14
 authusername, 33–146
 backoff, 33–101
 Defragmentation channel, 52–4
 ims-ms channel automatic override, 49–29
 ims-ms channels, 51–1
 LMTP client channel override via MAILBOX_BUSY_FAST_RETRY, 49–29
 bangonly, 33–34
 bangoverpercent, 33–34
 Rewrite rule address interpretation, 34–3, 34–4
 bangstyle, 33–35
 bidirectional, 33–102
 binaryclient, 33–117
 binaryserver, 33–117
 blocketrn, 33–116
 blocklimit, 33–112
 error_text_block_over MTA option, 39–159
 Notification messages, 47–25
 BSMTP, 33–51
 contchar, 33–51
 contposition, 33–51
 notick, 33–51
 tick, 33–52
 verb_never, 33–52
 verb_none, 33–52
 verb_off, 33–52
 verb_on, 33–52
 cacheeverything, 33–135
 cachefailures, 33–135
 cachesuccesses, 33–135
 caption, 33–56
 Changes take effect, 39–10
 Character sets and eight bit data, 33–52
 charset7, 33–52
 charset8, 33–52
 charsetesc, 33–52
 checkehlo, 33–118
 checkrrvs, 33–35, 33–118
 chunking*
 BURL interaction, 49–11
 chunkingclient, 33–119
 chunkingserver, 33–119
 binaryserver enables BDAT even without, 33–117
 clonehosts, 33–36, 33–61
 commentinc, 33–66
 commentmap, 33–66
 use_comment_strings MTA option, 39–198
 commentomit, 33–66
 commentstrip, 33–66
 commenttotal, 33–66
 conditionalpassthrough, 33–119
 conditionalrelay, 33–119
 conditionalsecuritymultipart, 33–45
 connectalias, 33–136
 connectcanonical, 33–136
 contchar, 33–51
 contposition, 33–51
 Conversion tags, 33–55
 convertoctetstream, 33–45
 copysendpost, 33–94, 47–1
 copywarnpost, 33–94, 47–1
 daemon, 33–61, 33–136
 Routing to a gateway, 49–47
 datefour, 33–66
 datetwo, 33–66
 dayofweek, 33–66
 defaulthost, 33–36, 33–67
 Use of value in rewrite rule substitution, 34–21
 defaultmx, 33–137
 defaultnameservers, 33–137
 deferralrejectlimit, 33–87, 33–120
 deferred, 33–102
 deferreddestination, 33–102
 deferredsource, 33–102
 defragment, 33–46, 52–3
 ims-ms channels, 51–1, 52–3
 LMTP channels, 52–3
 deletemessagehash, 33–90
 deliverbychannel, 33–124
 deliverbymin, 33–124
 deliveryflags, 33–108, 33–123
 dequeueremoveroute, 33–37
 description, 33–56
 destinationconversiontag, 33–55
 destinationdkimignore, 33–56
 destinationdkimpreserve, 33–56
 dkim_ignore_domains MTA option's effect on, 39–154

- dkim_preserve_domains MTA option's effect on, 39–155
- destinationdkimremove, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_remove_domains MTA option's effect on, 39–155
- destinationfilter, 33–109
 - Message capture example, 5–35
 - Sieve hierarchy, 5–65
- destinationnosolicit, 33–124
- destinationpassthrough, 33–119
- destinationrsrs, 33–29
- disabledestinationfilter, 33–109
- disableetrn, 33–116
- disablesourcefilter, 33–109
- disconnectbadauthlimit, 33–149
- disconnectbadburllimit, 33–87, 33–120, 49–11
- disconnectbadcommandlimit, 33–87, 33–120
- disconnectcommandlimit, 33–87, 33–120
- disconnectrecipientlimit, 33–87, 33–120
- disconnectrejectlimit, 33–87, 33–120
- disconnecttransactionlimit, 33–124
- Display label, 33–55
- dispositionchannel, 33–95, 47–22
- DKIM, 33–56
- dkimignore, 33–56
- dkimpreserve, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_preserve_domains MTA option's effect on, 39–155
- dkimremove, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_remove_domains MTA option's effect on, 39–155
- domainetrn, 33–116
- domainvrfy, 33–125
- dropblank, 33–68
- ehlo, 33–118
- eightbit, 33–53, 33–125
- eightnegotiate, 33–53, 33–125
- eightstrict, 33–53, 33–125
 - error_text_unnegotiated_eightbit MTA option, 39–166
- enqueueremoveroute, 33–37
- envelopetunnel, 33–68
- Error interpretation, 33–57
- errsendpost, 33–94, 47–1
- errwarnpost, 33–94, 47–1
- expandchannel, 33–58, 33–89, 33–103, 33–113
- expandlimit, 33–58, 33–89, 33–103, 33–113, 52–18
- expirysource, 33–68, 33–105
- explicitaslexternal, 33–151
- expnallow, 33–126
- expndefault, 33–126
- expndisable, 33–126
- exproute, 33–38
 - exproute_forward MTA option, 39–57
- externalidentity, 33–146
- File creation in the MTA queue area, 33–57
 - addrspfile, 33–57
 - expandchannel, 33–58, 33–89, 33–103, 33–113
 - expandlimit, 33–58, 33–89, 33–103, 33–113
 - multiple, 33–57
 - single, 33–57
 - single_sys, 33–57
 - subdirs, 33–59
- fileinto, 33–110
 - ims-ms channels, 33–110, 51–1
 - LMTP client (tcp_lmtpcs*) channels, 33–111
 - Subaddresses in addresses, 33–110
- filesperjob, 33–99
- filter, 33–109
 - Sieve hierarchy, 5–65
- fixsyntaxerrors, 33–68
- flagtransfer, 33–108, 33–123
 - Previously discarded message flag, 52–9
- forwardcheckdelete, 33–138
- forwardchecknone, 33–138
- forwardchecktag, 33–138
- Functional group list, 33–14
- futurerelease, 33–104, 33–126
- Gateway/firewall/mailhub, 33–59
 - aliasdetourhost, 33–30, 33–60
 - aliasoptindetourhost, 33–30, 33–60
 - daemon, 33–61, 33–136
 - lastresort, 33–62, 33–140
 - multigate, 33–62
 - nomultigate, 33–62
- generatemessagehash, 33–90
 - vnd.oracle.message-hash Sieve environment item, 5–15
- headerbottom, 33–69
- headercut, 33–69
- headerfoldpreserve, 33–69
- headerfoldremove, 33–69
- headerinc, 33–69
- headerkeeporder, 33–70
- headerlabelalignment, 33–69
- headerlimit, 33–71
- headerlineincrement, 33–69
- headerlinelength, 33–69
- headeromit, 33–69
- headerread, 33–70

- Header option file, 33–155
- Header option file, Location, 33–156, 33–156
 - test -rewrite utility, 58–35
- Headers, 33–62
- headerset7, 33–53
- headerset8, 33–53
- headersetesc, 33–53
- headertrailingpreserve, 33–72
- headertrailingremove, 33–72
- headertrim, 33–70
 - Header option file, 33–155
 - Header option file, Location, 33–156
 - Removing Received: header lines, 57–3
 - test -rewrite utility, 58–35
- header_733, 33–35
- header_822, 33–35
- header_uucp, 33–35
- holdlimit, 33–58, 33–89, 33–103, 33–113
 - Diagnosing .HELD files, 52–11
- Host names, 33–79
 - additional_host_names, 33–80
 - local_host_alias, 33–79
 - official_host_name, 33–79
- identnone, 33–138
- identnonelimited, 33–138
- identnonenumeric, 33–138
- identnonesymbolic, 33–138
- identtcp, 33–138
- identtcplimited, 33–138
- identtcpnumeric, 33–138
- identtcpsymbolic, 33–138
- ignoreencoding, 33–46
- ignoremessageencoding, 33–46
- ignoremultipartencoding, 33–46
- ignorerrvs, 33–35, 33–118
- implicitaslexternal, 33–151
- improute, 33–38
 - improute_forward MTA option, 39–57
- includefinal, 33–95
 - Recipient address reported in notifications, 47–6, 47–17
- Incoming channel match and switch, 33–81
- inner, 33–72
- innertrim, 33–70
 - Header option file, 33–155
 - Header option file, Location, 33–156
 - Removing Received: header lines, 57–3
 - test -rewrite utility, 58–35
- interfaceaddress, 33–139
- interpretencoding, 33–46
- interpretmessageencoding, 33–46
- interpretmultipartencoding, 33–46
- keepmessagehash, 33–90
- language, 33–72, 33–96
- lastresort, 33–62, 33–140
 - AUTH_ACCESS \$B flag, 49–39
- Length limits, 33–7
- limitheadertermination, 33–73
- linelength, 33–47
- linelimit, 33–112
 - error_text_line_over MTA option, 39–159
- lmtp, 33–127
- lmtp*
 - Imply nonotary, 33–96, 33–131
- lmtp_cr, 33–127
- lmtp_crlf, 33–127
- lmtp_crorlf, 33–127
- lmtp_lf, 33–127
- localbehavior, 33–39
- localvrfy, 33–125
- local_host_alias, 33–80
 - Overridden by BANNER_HOST, 49–21
 - Overridden by BANNER_REVERSE_HOST, 49–21
- logging, 33–84, 55–1
- Logging and debugging, 33–84
- logheader, 33–84
- Long address lists or headers, 33–86
- loopcheck, 33–128
- mailfromdnsverify, 33–129, 33–140
 - Bit in returnenvelope, 33–99
 - Bit in return_envelope, 39–156, 39–215
 - DNS verification, test -rewrite utility, 58–35
 - error_text_mailfromdnsverify MTA option, 39–165
- master, 33–102
- master_debug, 33–85
 - AUTH_ACCESS mapping \$U flag, 49–38
 - ims-ms channels, 51–6, 51–7
 - mm_debug MTA option, 39–74
 - os_debug MTA option, 39–75
 - Reprocess channel, 33–85, 52–19
- maxblocks, 33–48
- maxheaderaddrs, 33–73
- maxheaderchars, 33–73
- maxjobs, 33–99, 33–105
 - ims-ms channels, 51–1
 - Job Controller operation, 42–2
 - Modified effect under stress, 42–4
 - Modified under stress, stressjobs Job Controller option, 42–15
 - Modified under stress, unstressjobs Job Controller option, 42–15
- maxlines, 33–48
- maxperiodicnonurgent, 33–105
- maxperiodicnormal, 33–105

maxperiodicurgent, 33–105
maxprocchars, 33–90
maysasl, 33–149
maysaslclient, 33–149
 AUTH_ACCESS mapping, 49–38
maysaslserver, 33–149
maytls, 33–83, 33–151
maytlsclient, 33–83, 33–151
maytlsserver, 33–83, 33–151
 Should be set on SMTP SUBMIT server
 channel, 33–119
Message hash, 33–90
Message tracking, 33–91
 nottracking*, 33–91
 tracking*, 33–91
 trackinggenerate, 33–92
minperiodicnonurgent, 33–105
minperiodicnormal, 33–105
minperiodicurgent, 33–105
missingrecipientpolicy, 33–39, 33–73
MLS (Multi Layer Security), 33–93
mlslabel, 33–93
mlsrange, 33–93
msexchange, 33–49, 33–130, 33–152
mtprioritiesallowed, 33–106, 33–130
mtprioritiesrequired, 33–106, 33–130
multigate, 33–62
 LMTP, 39–95
multiple, 33–57
 Channel to a gateway system, 49–47
mustersasl, 33–149
mustersaslclient, 33–149
 AUTH_ACCESS \$G flag, 49–39
 AUTH_ACCESS mapping, 49–38
mustersaslserver, 33–149
 Required for implicitsaslexternal to take
 effect, 33–151
 Should be set on SMTP SUBMIT server
 channel, 33–119
musttls, 33–83, 33–151
musttlsclient, 33–83, 33–151
 AUTH_ACCESS \$T flag, 49–39
musttlsserver, 33–83, 33–151
mx, 33–137
 AUTH_ACCESS mapping table \$M flag,
 49–39
 AUTH_ACCESS mapping table \$X flag, 49–39
nameparameterlengthlimit, 33–49
nameservers, 33–137
 DNS verification, 33–138
 Reverse lookups, 33–138
noaddlineaddr, 33–29
noaddresssrs, 33–29
noaddrreturnpath, 33–63
nobangorpercent, 33–34
nobangoverpercent, 33–34
 Rewrite rule address interpretation, 34–4
nobinaryclient, 33–117
nobinaryserver, 33–117
noblocklimit, 33–112
nocache, 33–135
nochunking*
 BURL interaction, 49–11
nochunkingclient, 33–119
nochunkingserver, 33–119
noconvertoctetstream, 33–45
nodayofweek, 33–66
nodefaulthost, 33–36, 33–67
nodefragment, 33–46
nodestinationfilter, 33–109
nodestinationsrs, 33–29
nodns, 33–137
nodropblank, 33–68
noehlo, 33–118
noexpirysource, 33–68, 33–105
noexproute, 33–38
nofileinto, 33–110
nofilter, 33–109
noflagtransfer, 33–108, 33–123
noheaderread, 33–70
 Header option file, 33–155
noheadertrim, 33–70
 Header option file, 33–155
noimproute, 33–38
noinner, 33–72
noinnertrim, 33–70
 Header option file, 33–155
nolinelimit, 33–112
nolocalbehavior, 33–39
nologging, 33–84
noloopcheck, 33–128
nomailfromdnsverify, 33–129, 33–140
nomaster_debug, 33–85
nomsexchange, 33–49, 33–130, 33–152
nomultigate, 33–62
nomx, 33–137
nonotary, 33–96, 33–131
nonrandommx, 33–137
nonurgentafter, 33–100
nonurgentbackoff, 33–101
nonurgentblocklimit, 33–114
nonurgentnotices, 33–96
noreceivedfor, 33–74
 Limiting emission of internal host names,
 57–3
noreceivedfrom, 33–74

- Limiting emission of internal host names, 57–3
- noremotehost, 33–36, 33–67
- norestricted, 33–40
- noreturnaddress, 33–97
- noreturnpersonal, 33–97
- noreverse, 33–41
- normalafter, 33–100
- normalbackoff, 33–101
- normalblocklimit, 33–114
- normalnotices, 33–96
- norules, 33–41
- nosasl, 33–149
- nosaslclient, 33–149
- nosaslpassauth, 33–154
- nosaslserver, 33–149
- nosaslswitchchannel, 33–82, 33–154
- nosasltrustauth, 33–154
- nosendetn, 33–131
- nosendpost, 33–94, 47–1
- noserviceconversion, 33–55
- noslave_debug, 33–85
- nosocks, 33–142
- nosourcefilter, 33–109
- nosourcesrs, 33–29
- nosubdirs, 33–59
- noswitchchannel, 33–81
- notary, 33–96, 33–131
- nothurman, 33–50
- notices, 33–96
 - Defragmentation channel, 52–4
 - filter_discard channel, 52–7
 - ims-ms channels, 51–1
 - Local channel value, 52–2
 - Notification message generation, 47–4
 - return_units MTA option, 39–216
- notick, 33–52
- Notification messages and postmaster messages, 33–93
- notificationchannel, 33–95, 47–22
- notls, 33–83, 33–151
- notlsclient, 33–83, 33–151
- notlsserver, 33–83, 33–151
- notracking*, 33–91
- noturn, 33–132
- novrfy, 33–125
- nowarnpost, 33–94, 47–1
- noxclient, 33–76, 33–132, 33–153
- nox_env_to, 33–75
- official_host_name, 33–79
 - Domain used to construct message-id's, id_domain MTA option overrides, 39–221
 - ims-ms channels, 51–1
 - L channel's name used communicating with remote hosts, Overridden by BANNER_HOST, 49–21
 - L channel's name used communicating with remote hosts, Overridden by local_host_alias, 33–80
 - Overridden by ldap_default_domain, 39–222
 - Overridden by local_host_alias, 33–80
 - Overridden by received_domain, 39–222
- parameterformatdefault, 33–50, 33–54
- parameterformatminimizeencoding, 33–50, 33–54
- parameterformatstripencoding, 33–50, 33–54
- parameterlengthlimit, 33–49
- passyntaxerrors, 33–68
- passthrough, 33–119
 - destinationdkim* trigger, 33–56
 - dkimpreserve trigger, 33–56
 - dkim_ignore_domains avoids triggering, 39–154
 - dkim_preserve_domains trigger, 39–155
- percentonly, 33–34
- percents, 33–35
- personalinc, 33–41, 33–76
- personalmap, 33–41, 33–76
 - use_personal_names MTA option, 39–201
- personalomit, 33–41, 33–76
- personalstrip, 33–41, 33–76
- pool, 33–106
 - ims-ms channels, 51–1
 - Job Controller operation, 42–2
 - job_pool Job Controller option, 42–17
- port, 33–139, 33–142
 - AUTH_ACCESS mapping table \$P flag, 49–38
- postheadbody, 33–98
- postheadonly, 33–98
- Processing control and job submission, 33–99
- processsecuritymultiparts, 33–45
- randommx, 33–137
- readreceiptmail, 33–98
- receivedfor, 33–74
- receivedfrom, 33–74
- receivedstate, 33–77
 - Conversion channel, 38–5
 - filter_discard channel, 52–7
- recipientcutoff, 33–87, 33–120
- recipientlimit, 33–87, 33–120
 - error_text_recipient_over MTA option, 39–161
- refuseehlo, 33–118
- refusenotary, 33–96, 33–131
- rejectsmtpplonglines, 33–133
 - error_text_smtp_lines_too_long MTA option, 39–166

- relaxheadertermination, 33–73
- relay, 33–119
- remotehost, 33–36, 33–67
- reportboth, 33–98
- reporthead, 33–98
- reportnotary, 33–98
- reportsuppress, 33–98
- restricted, 33–40
 - restricted switch of test -rewrite utility, 58–37
- retainsecuritymultiparts, 33–45
- returnaddress, 33–97
- returnenvelope, 33–99
 - DNS verification, test -rewrite utility, 58–35
 - error_text_invalid_return_address MTA option, 39–166
 - error_text_mailfromdnsverify MTA option, 39–165
 - error_text_unknown_return_address MTA option, 39–166
 - return_envelope MTA option, 39–156, 39–216
- returnpersonal, 33–97
 - return_personal MTA option, 39–216
- reverse, 33–41
- routelocal, 33–42
 - Compared to localbehavior, 33–39
 - Removal of source routes during rewriting, 34–8
- Routing
 - aliasdetourhost, 33–30, 33–60
 - aliasoptindetourhost, 33–30, 33–60
 - daemon, 33–61, 33–136
 - lastresort, 33–62, 33–140
 - multigate, 33–62
 - nomultigate, 33–62
- rules, 33–41
 - Source channel-specific rewriting, 34–26
- SASL and TLS, 33–146
- saslpassauth, 33–154
- saslruleset, 33–154
- saslswitchchannel, 33–82, 33–154
 - Effect nullified by XUNAUTHENTICATE SMTP command, 33–83, 33–155
 - SMTP relay blocking, 49–48
- sasltrustauth, 33–154
- scriptlimit, 33–111
- secondclassafter, 33–100
- secondclassblocklimit, 33–114
- sendetrn, 33–131
- sendpost, 33–94, 47–1
- Sensitivity limits, 33–107
- sensitivitycompanyconfidential, 33–107
- sensitivitynormal, 33–107
- sensitivitypersonal, 33–107
- sensitivityprivate, 33–107
- Service conversions, 33–55
- serviceconversion, 33–55
- sevenbit, 33–53, 33–125
- Sieve filters
 - *flagtransfer, 33–108, 33–123
- Sieve filters and delivery flags, 33–108
 - *addrtypescan*, 33–29, 33–108
 - *filter, 33–109
 - deliveryflags, 33–108, 33–123
 - fileinto, 33–110
 - nofileinto, 33–110
 - scriptlimit, 33–111
- silentetrn, 33–116
- single, 33–57
 - Channel to a gateway system, 49–47
 - Effect via deliveryflags channel option, 33–108, 33–123
 - Pipe channels, 52–13, 52–16
- single_sys, 33–57
 - Channel to a gateway system, 49–47
- Size limits on messages, 33–111
- slave, 33–102
- slave_debug, 33–85
 - \$U flag in address *_ACCESS mapping table, 44–9
 - \$U flag in PORT_ACCESS mapping table, 44–4
 - mm_debug MTA option, 39–74
 - os_debug MTA option, 39–75
 - RESETDEBUG Archive option, 45–9
- smtp, 33–127
- SMTP and LMTP protocol
 - deliverbychannel, 33–124
- SMTP protocol, 33–116
- smtp_cr, 33–127
- smtp_crlf, 33–127
- smtp_crorlf, 33–127
- smtp_lf, 33–127
- sockshost, 33–142
- socksnoauth, 33–142
- sockspassword, 33–142
- socksport, 33–142
- socksusername, 33–142
- socksuserpassword, 33–142
- sourceblocklimit, 33–112
- sourcecommentinc, 33–66
- sourcecommentmap, 33–66
 - use_comment_strings MTA option, 39–198
- sourcecommentomit, 33–66
- sourcecommentstrip, 33–66
- sourcecommenttotal, 33–66
- sourceconversiontag, 33–55

sourcefilter, 33–109

- error_text_source_sieve_access MTA option, 39–166
- error_text_source_sieve_authorization MTA option, 39–166
- error_text_source_sieve_syntax MTA option, 39–166
- Message capture example, 5–35
- Sieve hierarchy, 5–65

sourcenosolicit, 33–124

sourcepersonalinc, 33–41, 33–76

sourcepersonalmap, 33–41, 33–76

- use_personal_names MTA option, 39–201

sourcepersonalomit, 33–41, 33–76

sourcepersonalstrip, 33–41, 33–76

sourceroute, 33–34

sourcesrs, 33–29

Spam/virus filter package use, 33–115

spfhelo, 33–143

spfmailfrom, 33–143

spfnone, 33–143

spfrcptto, 33–143

streaming, 33–134

subaddressexact, 33–42

- Address reversal, 35–55

subaddressrelaxed, 33–42

- Address reversal, 35–55
- Subaddresses on aliases, 35–45

subaddresswild, 33–42

- Address reversal, 35–55

subdirs, 33–59

submit, 33–119

suppressfinal, 33–95

- Recipient address reported in notifications, 47–6, 47–17

switchchannel, 33–81

- Effectively disabled if CHECK_SOURCE=0, 49–22
- INTERNAL_IP mapping table, 44–6
- SMTP relay blocking, 49–48

TCP/IP connections and DNS lookups, 33–135

thirdclassafter, 33–100

threaddepth, 33–106, 33–146

- Channel to a gateway system, 49–47
- Job Controller operation, 42–2
- Modified under stress, stressfactor Job Controller option, 42–15
- Modified under stress, unstressfactor Job Controller option, 42–15

thurman, 33–50

tick, 33–52

tlsswitchchannel, 33–83, 33–151

tracking*, 33–91

trackinggenerate, 33–92

transactionlimit, 33–124

- error_text_transaction_limit_exceeded MTA option, 39–166
- Reprocess channel, 52–19

truncatesmtplonglines, 33–133

turn, 33–132

turn_in, 33–132

turn_out, 33–132

unrestricted, 33–40

urgentafter, 33–100

urgentbackoff, 33–101

urgentblocklimit, 33–114

urgentnotices, 33–96

useintermediate, 33–95

- Recipient address reported in notifications, 47–6, 47–17

usepermanenterror, 33–57

user, 33–62, 33–107

- Pipe channels, 52–14
- See also pipeuser option in restricted.cnf, 33–62, 33–107

usereplyto, 33–78

useresent, 33–78

userswitchchannel, 33–81

- Address reversal, 35–50

usetemporaryerror, 33–57

utf8strict

- error_text_unnegotiated_eightbit MTA option, 39–166

Values

- Case preservation, 33–7
- Length limits, 33–7
- Quoting, 33–7
- URL types, 1–4

verb_never, 33–52

verb_none, 33–52

verb_off, 33–52

verb_on, 33–52

viaaliasoptional, 33–44

viaaliasrequired, 33–44

- Critical that it be set on the local channel, 52–1
- Error text if user not found, 39–158
- ims-ms channels, 51–3

vrfyallow, 33–134

vrfydefault, 33–134

vrfyhide, 33–134

warnpost, 33–94, 47–1

wrapsmtplonglines, 33–133

xclient, 33–76, 33–132, 33–153

xclientrepeat, 33–76, 33–132, 33–153

xclientsasl, 33–76, 33–132, 33–153

xclientsaslrepeat, 33–76, 33–132, 33–153

x_env_to, 33–75

Channels

- "I"
 - See Local channel, 52–1
- Available, 33–2
- Bitbucket
 - See Bitbucket channel, 52–2
- BSMTP
 - See BSMTP channels, 50–1
- Configuration, 33–5
- Conversion
 - See Conversion channel, 38–1
- Defragment
 - See Defragmentation channel, 52–3
- Defragmentation
 - See Defragmentation channel, 52–3
- Executable program
 - master_command Job Controller option, 42–11
 - slave_command Job Controller option, 42–14
- filter_discard
 - See filter_discard channel, 52–7
- Generic SMTP
 - See Generic SMTP channels, 52–9
- Hold
 - See Hold channel, 52–10
- ims-ms
 - See ims-ms channels, 51–1
- List of, 33–2
- LMTP
 - See also TCP/IP channels, 49–2
 - See LMTP channels, 49–2
- Local
 - See Local channel, 52–1
- Master program, 33–1
- Name used by SMS gateway for enqueueing
 - mta_channel gateway_profile option, 53–5
- Overview, 33–1
- Pipe
 - See Pipe channel, 52–12
- Process
 - Process channel, 52–18
- Reprocess
 - Reprocess channel, 52–18
- Reserved names, 33–2
- Sieve filter
 - See also destinationfilter channel option, 33–109
 - See also sourcefilter channel option, 33–109
 - vnd.sun.destination-channel environment item, 5–15
 - vnd.sun.source-channel environment item, 5–15, 5–15
- Slave program, 33–1
- SMTP over TCP/IP
 - See TCP/IP channels, 49–2
- Switch of effective source
 - \$S flag in PORT_ACCESS mapping table, 44–4
 - *switchchannel channel options, 33–81
 - Adding domain name to "bare" username, 33–37, 33–67
 - Does not effect SMTP line terminator selection, 33–128
 - saslswitchchannel channel option, 33–82, 33–154
 - tlsswitchchannel channel option, 33–84, 33–152
- TCP/IP
 - See TCP/IP channels, 49–2
- test_smtp
 - See Generic SMTP channels, 52–9
- channel_class group, 42–17
- Character set
 - Authentication
 - broken_client_login_charset auth option, 12–2
 - charset parameter in return_prefix.txt file must match charset used in return_option.opt values, 47–14
 - charset7 channel option, 33–52
 - charset8 channel option, 33–52
 - charsetesc channel option, 33–52
 - Conversion, 38–1, 38–18
 - Allowed character sets, 33–52
 - CHARSET-CONVERSION mapping table, 38–16
 - translate Sieve filter action, 5–62
 - detectcharset MSHTTP option, 29–4
 - headerset7 channel option, 33–53
 - headerset8 channel option, 33–53
 - headersetesc channel option, 33–53
 - httpcharset MSHTTP option, 29–14
 - Initial (incoming) character set labelling
 - charset* channel options, 33–52
 - ISO-2022-JP
 - charsetesc channel option, 33–53
 - ISO-2022-KR
 - charsetesc channel option, 33–53
 - Japanese
 - ISO-2022-JP, charsetesc channel option, 33–53
 - Korean
 - ISO-2022-KR, charsetesc channel option, 33–53
 - Line wrap and encoding interaction, 33–134
 - mailcharset MSHTTP option, 29–14
 - Messenger Express

- rfc822headerallow8bit base option, 7–15
- MIME parameter
 - RFC 2231 encoding removal, 33–51, 33–54
- MSHTTP validation
 - charsetvalidation MSHTTP option, 29–4
- Sieve filters
 - translate function, 5–62
 - utf-8, 5–13
- smsc_default_charset gateway_profile option, 53–8
- UNKNOWN, 33–53
- Used by SMS gateway for enqueued message body
 - email_body_charset gateway_profile option, 53–5
- Charset
 - See Character set, 38–1
- charset7 channel option, 33–52
- charset8 channel option, 33–52
- charsetesc channel option, 33–52
- charsetvalidation MSHTTP option, 29–4
- checkdiskusage Message Store option, 16–5
- checkehlo channel option, 33–118
- checkinterval Message Store deadlock option, 16–17
- checkmailhost Message Store option, 16–5
- checkoverssl smime option, 30–5
- Checkpointing
 - Message Store operation, 16–16
 - Message transmission, 33–48
- checkrrvs channel option, 33–35, 33–118
- check_memcache.so, 37–24
- check_metermaid.so, 37–27
 - Example of use, 49–45, 55–13, 55–15
- chunking* channel options
 - BURL interaction, 49–11
- chunkingclient channel option, 33–119
- chunkingserver channel option, 33–119
 - binaryserver enables BDAT even without, 33–117
- chunk_cache_limit MTA option, 39–177
- CIDR notation, G–2
 - Mapping table wildcards, 37–6
 - TCP wrapper filter wildcard patterns, 6–4
- Cipher suites
 - ssladjustciphersuites option, 7–11, 27–24
- Circuit check
 - circuitcheck_completed_bins MTA option, 39–70
- Counters
 - Binning of, circuitcheck_completed_bins MTA option, 39–70
 - MTA options
 - circuitcheck_completed_bins, 39–70
 - circuitcheck_completed_bins MTA option, 39–70
 - circuitcheck_paths_size MTA option, 39–177
- ClamAV
 - See Spam/virus filter package integration, ClamAV, 45–4
- cleanonly Message Store deleted expire option, 16–27
- cleanup Message Store option, 16–10
- clientlookup submitproxy/vdomain option, 27–9
- clonehosts channel option, 33–36, 33–61
- cmapldapattr Base certmap option, 7–24
- cmapldapattr certmap option, 7–24
- command pipe option, 52–15
- Comment lines
 - comment_chars MTA option, 39–171
 - MTA option file, 39–9
 - comment_chars MTA option does not affect, 39–172
- commentinc channel option, 33–66
- commentmap channel option, 33–66
 - use_comment_strings MTA option, 39–198
- commentomit channel option, 33–66
- commentstrip channel option, 33–66
- commenttotal channel option, 33–66
- comment_chars MTA option, 39–171
 - Alias database, 35–44
 - Domain database, 34–35
 - General database, 37–21
- Communications Express
 - Address search
 - allowldapaddresssearch MSHTTP option, 29–4
- Compiled configuration
 - Testing of
 - test -rewrite utility, 58–28
- compliance Message Store archive option, 16–14
- conditionalpassthrough channel option, 33–119
- conditionalrelay channel option, 33–119
- conditionalsecuritymultipart channel option, 33–45
- configutil parameters
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - local.hostname, 39–84, 39–98
 - Direct LDAP alias lookups, 35–6
 - local.imta.hostnamealiases
 - Direct LDAP alias lookups, 35–6
 - MTA use, 39–84, 39–98
 - local.imta.mailaliases
 - Direct LDAP alias lookups, 35–6
 - local.imta.schematag
 - Direct LDAP alias lookups, 35–6

- Direct LDAP domain lookups, 34–30
- local.ldapconnecttimeout
 - Direct LDAP alias lookups, 35–5
- local.ldapsearchtimeout
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- local.service.pab.ldapbinddn, 39–183
- local.service.pab.ldapghost, 39–183
- local.service.pab.ldappasswd, 39–183
- local.service.pab.ldapport, 39–183
- local.ugldapbasedn
 - Direct LDAP alias lookups, 35–6
- local.ugldapbindcred
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- local.ugldapbinddn
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- local.ugldapghost
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- local.ugldapport
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- local.ugldapusessl
 - Direct LDAP alias lookups, 35–5
- logfile.imta.syslogfacility
 - Affect on log_messages_syslog, 39–249
- service.dccroot
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
- service.defaultdomain
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
- config_debug MTA option, 39–73
- connectalias channel option, 33–136
- connectcanonical channel option, 33–136
- Connection access control
 - PORT_ACCESS mapping table, 44–2
- connecttimeout IMAP Proxy option, 27–10
- connecttimeout indexer option, 25–2
- connecttimeout MeterMaid Client option, 46–4
- connecttimeout MMP option, 27–10
- connecttimeout POP Proxy option, 27–10
- connlimits MSHTTP IMAP POP MMP Proxy option, 27–10
- connrejectthreshold MMP option, 27–11
- contchar channel option, 33–51
- contenttype Message Store message type mtindex option, 16–21
- content_return_block_limit MTA option, 39–207, 39–213
- contextname SNMP option, 60–2
- Continuation lines
 - In aliases file, 35–24
- contposition channel option, 33–51
- Conversion channel, 38–1
 - \$R input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148
 - Alternate routing, 38–3
 - Configuration, 38–5
 - Control of conversion operation, 38–6
 - Conversion entries
 - Backslash quoting, 38–12
 - Environment variables, 38–12
 - Mapping table callouts, 38–14
 - Parameters, 38–8
 - Parameters, Wildcards in values, 38–11
 - Single quote character, 38–12
 - Symbol substitution, 38–12
 - Symbols, 38–12
 - Conversion scripts
 - Exit statuses, 38–15
 - Header access, 38–14
 - Conversion tag, 38–3
 - Conversion tags
 - Adding via address access mapping tables, 44–9
 - CONVERSIONS mapping table, 38–2
 - Conversion tag, 38–3
 - Entry scanning and application, 38–7
 - ignore*encoding channel options, 33–47
 - receivedstate channel option, 38–5
 - Conversion tags, 38–15, G–3
 - *conversiontag Sieve actions, 5–18, 5–48
 - tag switch of test -rewrite utility, 58–38
 - Address reversal, 35–50
 - alias_conversion_tag alias option, 35–11
 - Channel options, 33–55
 - CHARSET-CONVERSION mapping table, 38–17
 - CONVERSIONS mapping table, 38–3
 - include_conversiontag MTA option, 38–2
 - CONVERSION_TAG alias file named parameter, 35–30
 - deliveryflags channel option, 33–109, 33–123
 - destinationconversiontag channel option, 33–55
 - Domain
 - ldap_domain_attr_conversion_tag MTA option, 39–146
 - ldap_domain_attr_source_conversion_tag MTA option, 39–146, 39–146
 - FORWARD mapping table probes, 35–59
 - include_conversiontag MTA option, 39–190
 - ldap_conversion_tag MTA option, 39–123
 - ldap_source_conversion_tag MTA option, 39–120

- Logging of
 - log_conversion_tag MTA option, 39–255
- MESSAGE-SAVE-COPY mapping table probe, 54–4, 54–5
- message_save_copy_flags MTA option, 39–198
- See also Sieve filters, Conversion tags, 5–48
- Sieve filter access to, 5–26
- Sieve filters, 5–48
- sourceconversiontag channel option, 33–55
- TAG conversion entry parameter, 38–10
- Conversions, 38–1
 - MIME relabelling, 38–25
 - RELABEL conversion entry parameter, 38–8
 - Service conversions, 38–27
 - Channel options, 33–55
 - serviceconversion channel option, 33–55
- conversions file, 38–1
- conversions MTA option, 38–1, 39–69
- conversion_size MTA option, 39–177
- convertoctetstream channel option, 33–45
- cookiedomain MSHTTP option, 29–8
- cookie_name MSHTTP option, 29–4
- copymsg_notifytarget option, 24–6
- copysendpost channel option, 33–94, 47–1
- copywarnpost channel option, 33–94, 47–1
- count Message Store purge option, 16–23
- Counters
 - See MTA options, 39–69
- CRAM-MD5, G–3
 - crams mmp/imaproxy/popproxy/vdomain option, 27–11
 - has_plain_passwords auth option, 12–1
 - userPassword LDAP attribute
 - Contain clear-text password, 39–103
- crams MMP/IMAP Proxy/POP Proxy/Virtual Domain option, 27–11
- CRL
 - See Certificate, Revocation List, 30–4
- crl_dir smime option, 30–4
- crl_enable smime option, 30–4
- crl_mappingurl smime option, 30–5, 30–5
- crlurl_login_dn smime option, 30–4
- crlurl_login_pw smime option, 30–5
- crlusepastnextupdate smime option, 30–7
- crontab Scheduler task option, 8–2
- crontab Scheduler task:expire option, 8–3, 16–18
- crontab Scheduler task:msprobe option, 8–3, 10–2
- crontab Scheduler task:purge option, 8–3, 16–24
- crontab Scheduler task:return_job option, 8–3, 8–4, 47–4
- crontab Scheduler task:snapshot option, 8–4, 8–5
- crontab Scheduler task:snapshotverify option, 8–4, 8–5

D

- daemon channel option, 33–61, 33–136
 - Routing to a gateway, 49–47
- data: URLs
 - data:, discard;
 - spamfilterN_null_action options' default value, 39–238
 - data:, require "fileinto"; fileinto "\$U";
 - spamfilterN_string_action options' default value, 39–239
 - data:,\$M
 - spamfilterN_string_action option value for Milters, 39–240, 45–7, 45–14
 - Example of spamfilterN_string_action value, 45–3
 - Example spamfilter2_string_action option value, 45–9
 - MTA URL types, 1–4
- Database MTA options, 39–71
 - name_table_name (OpenVMS only), 39–59
- Databases
 - Defragment database used by defragmentation channel, 52–3
 - Job Controller queue cache, 42–1
 - cache -sync utility, 58–6
 - cache -walk utility, 58–8
 - max_cache_messages Job Controller option, 42–12
 - Operation under stress, 42–3
 - queue_cache_mode MTA option, 39–174
 - queue_cache_mode_3_files MTA option, 39–174
 - synch_time Job Controller option, 42–15
- LDAP
 - Aliases and domains stored in, 35–3
 - Aliases stored in, 35–5
- Memcache, 39–202
- MeterMaid, 39–211
- MTA alias database, 35–43
- MTA Forward database, 35–61
- MTA General database, 37–20
- MTA options, 39–71
- MTA Pipe database, 52–15
- MTA profile database, 52–15
- MTA Reverse database, 35–50
- Sieve extlists extension, 5–29

- data_type local_table MeterMaid option, 46–2
- datefour channel option, 33–66
- datetwo channel option, 33–66
- dayofweek channel option, 33–66
- da_host MSHTTP option, 29–7
- da_port MSHTTP option, 29–8

dbcachesize Message Store msghash option, 16–22
 dbcachesize Message Store option, 16–10
 dbblockcount base option, 7–10
 dblogregionmax Message Store option, 16–10
 dbnumcaches Message Store option, 16–5
 dbpriority Message Store dbreplicate option, 16–16
 dbregionmax Message Store option, 16–10
 dbremotehost Message Store dbreplicate option, 16–16, 16–17
 dbsync Message Store option, 16–5
 dbtmpdir Message Store option, 16–11
 dbtxnsync base option, 7–9
 dcroot base option, 7–16

- Direct LDAP alias lookups, 35–6
- Direct LDAP domain lookups, 34–30

 deadlockaggressive Message Store option, 16–5
 debug Base domainmap option, 7–3, 7–22
 debug Deployment Map option, 7–21, 14–1
 debug Dispatcher option, 7–22, 41–3
 debug domainmap option, 7–22
 debug Job Controller option, 7–22, 42–10
 debug Message Store checkpoint option, 7–22, 16–16
 debug MeterMaid Client option, 7–22, 46–2
 debug msadmin option, 7–21
 debug pmxbl option, 7–21
 debug sms_gateway option, 7–21, 53–2
 Debugging

- Address parsing by the MTA
 - ap_debug MTA option, 39–73
- Archive package integration
 - DEBUG Archive option, 45–9
 - RESETDEBUG Archive option, 45–9
- Authentication
 - \$A flag in PORT_ACCESS mapping table, 44–4
 - AUTH_DEBUG TCP/IP-channel-specific option, 49–20
 - AUTH_DEBUG TCP/IP-channel-specific option, 49–20
- cache_debug MTA option, 39–73
- Channel dequeue, 33–85
 - \$U flag in AUTH_ACCESS mapping, 49–38
 - dequeue_debug MTA option, 39–73
- Channel enqueue, 33–85
 - \$U flag in address *_ACCESS mapping table, 44–9
 - \$U flag in PORT_ACCESS mapping table, 44–4
 - mm_debug MTA option, 39–74
- Channel log files
 - ims-ms channel, 51–6
 - ims-ms channels, 33–86
 - LMTP server, 33–86
 - Reprocess channel, 33–85, 52–19
 - TCP/IP channels, 33–85
- Channel operation
 - See Debugging, Channel dequeue, 33–85
 - See Debugging, Channel enqueue, 33–85
- ClamAV
 - DEBUG ClamAV option, 45–4
- debugkeys option, 7–4, 27–12
 - os_debug MTA option, 39–75
- Dispatcher
 - debug Dispatcher option, 7–22, 41–3
 - debug_flush MTA option, 39–73, 39–172
- Flushing to disk
 - debug_flush MTA option, 39–73, 39–172
- Force via address access mapping tables, 44–9
- HULA
 - \$A flag in PORT_ACCESS mapping table, 44–4
- ICAP
 - DEBUG ICAP option, 45–5
- ims-ms and LMTP server
 - messagetrace.activate option, 23–1
- ims-ms channels, 51–7
- Job Controller
 - cache -walk utility, 58–8
 - debug Job Controller option, 7–22, 42–10
 - debug_flush MTA option, 39–73, 39–172
 - Enabling, imsimta cache -change -global -debug=N, 58–4
 - Example of enabling, 58–5
- LDAP
 - debugkeys option's ldap key, 27–12
 - ldaptrace base option, 7–10
- LDAP lookup cache
 - cache_debug MTA option, 39–73
- LDAP lookups
 - mm_debug MTA option, 39–74
- lpool
 - os_debug MTA option, 39–75
- master_debug channel option, 33–85
- Message Store checkpoint, 7–22, 16–16
- Message tracking
 - tracking_debug MTA option, 39–75
- Milter
 - DEBUG Milter option, 45–5
 - RESETDEBUG Milter option, 45–5
- mm_debug MTA option, 39–74
- MSHTTP
 - nofilecache MSHTTP option, 29–5
- MTA options, 39–71
- pmxbl callout
 - debug option, 7–21

- return_debug MTA option, 39–75
- return_job
 - return_debug MTA option, 39–75
 - return_verify MTA option, 39–75
- Sieve filter processing
 - mm_debug MTA option, 39–74
- Sieve filter processing (low-level)
 - filter_debug MTA option, 39–74, 39–231
- slave_debug channel option, 33–85
- SMS gateway
 - debug option, 7–21, 53–2
 - foreground sms_gateway option, 53–3
- SMTP server processes
 - debug_flush MTA option, 39–73, 39–172
- Spam/virus filter package integration
 - mm_debug MTA option, 39–74
- SpamAssassin
 - DEBUG SpamAssassin option, 45–7
- SPF lookups
 - mm_debug MTA option, 39–74
- TRACE_LEVEL TCP/IP-channel-specific option, 49–36
- debugkeys base option, 7–4, 27–12
 - os_debug MTA option, 39–75
- debugkeys mmp/imaproxy/popproxy/
submitproxy/vdomain option, 27–11
- decode_encoded_words MTA option, 39–224
- defaultacl Message Store deleted option, 16–26
- defaultdomain base option, 7–4, 27–13
 - Default for contextname SNMP option, 60–2
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
 - ims-ms channels, 51–4
- defaultdomain MMP/IMAP Proxy/POP Proxy/
SUBMIT Proxy/vdomain option, 27–12
- defaultdomain option, 27–12
- defaultdomain SUBMIT Proxy option, 27–12
- defaultdomain vdomain option, 27–12
- defaultthost channel option, 33–36, 33–67
 - use of value in rewrite rule substitution, 34–21
- defaultthostindex PAB option, 59–1
- defaultmx channel option, 33–137
- defaultnameservers channel option, 33–137
- defaultpartition Message Store option, 16–11
- deferralrejectlimit channel option, 33–87, 33–120
- deferred channel option, 33–102
- deferreddestination channel option, 33–102
- deferredsource channel option, 33–102
- defer_group_processing MTA option, 39–184
 - Mass mailings, 36–21
- defer_header_addition MTA option, 39–224
 - Sieve redirect action, 5–41
- defragment channel option, 33–46, 52–3
 - ims-ms channels, 51–1, 52–3
 - LMTTP channels, 52–3
- Defragmentation channel, 52–3
 - backoff channel option, 52–4
 - Configuration, 52–3
 - Defragment database, 52–3
 - Multi-host access, 52–3, 52–5
 - NFS storage, 52–3, 52–5
 - Fragment retention time, 52–4
 - Multi-host operation
 - Defragment database shared among hosts, 52–3, 52–5
 - Example, 52–6
 - notices channel option, 52–4
 - Options
 - MAX_PARTS, 52–4
- deleted Message Store expirerule option, 16–19
- deletemessagehash channel option, 33–90
- deletemsg notifytarget option, 24–4
- delimiter_char MTA option, 39–57
- deliverbychannel channel option, 33–124
- deliverbymin channel option, 33–124
- Delivery flags
 - Logging
 - log_delivery_flags MTA option, 39–262
- Delivery options
 - Testing of
 - test -rewrite utility, 58–28
- deliveryflags channel option, 33–108, 33–123
- delivery_options MTA option, 39–92
 - Effect on mailRoutingHosts interpretation, 39–144
- Denial-of-service
 - Defending against attacks, 44–17
 - Dispatcher self-protection features, 41–2
 - Job Controller self-protection features, 42–3
 - MMP
 - ldappendingoplimit, 27–16
 - stressfdwait base option, 7–5
 - stressperiod base option, 7–5
 - preauth mmp/imaproxy/popproxy/vdomain
option, 27–20
- Deployment Map, 1
 - Options, 14–1
 - capability_starttls, 14–2
 - debug, 7–21, 14–1
 - enable, 14–1
 - heartbeat, 14–1
 - port, 14–1
 - run_as_server, 14–2
 - server_host, 14–1
 - userid, 14–2
- Server

- msprobe probe of, 10–2
- dequeueeremoveroute channel option, 33–37
- dequeue_debug MTA option, 39–73
 - os_debug MTA option, 39–75
- dequeue_map MTA option, 39–172
- describe_cache_limit MTA option, 39–177
- description alarm.system:diskavail option, 11–2
- description alarm.system:serverresponse option, 11–2
- description channel option, 33–56
- destination Message Store archive option, 16–14
- destinationconversiontag channel option, 33–55
- destinationdkimignore channel option, 33–56
- destinationdkimpreserve channel option, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_preserve_domains MTA option's effect on, 39–155
- destinationdkimremove channel option, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_remove_domains MTA option's effect on, 39–155
- destinationfilter channel option, 33–109
 - Message capture example, 5–35
 - Performance impact, 56–3
 - Sieve hierarchy, 5–65
- destinationnosolicit channel option, 33–124
- destinationpassthrough channel option, 33–119
- destinationrsrs channel option, 33–29
- detectcharset MSHHTTP option, 29–4
- Dial up connections
 - SMTP ETRN extension, 49–51
- Dictionary attack
 - Blocking via LOG_ACTION, 55–19
 - Warning of possible, 55–13
- digest_on MTA option, 39–184
- Direct LDAP lookups
 - Address reversal
 - Caching, 39–153
 - domain_uplevel MTA option, 39–81
 - Performance tuning, 39–153
 - Aliases and addresses
 - configutil parameters, 35–5
 - Deferring group (list) expansion, defer_group_processing MTA option, 39–184
 - domain_uplevel MTA option, 39–81
 - Forwarding user mail, 35–59
 - MTA options, 35–5
 - Caching, 39–152
 - Domain lookups, domain_match_cache_size, 39–152
 - Domain lookups,
 - domain_match_cache_timeout, 39–152
 - Domains, 7–5, 34–30, 39–83, 39–153
 - Domains, domain_match_cache_size MTA option, 39–152
 - Domains, domain_match_cache_timeout MTA option, 39–152
 - Domains, ldap_domain_timeout MTA option, 39–152
 - Reverse addresses, 39–153
- Configuration of, 35–3
- Domains
 - Aliased domains, 39–144
 - Aliases for domains, domain_uplevel MTA option, 39–81
 - Caching, 7–5, 34–30, 39–83, 39–153
 - configutil parameters, 34–29
 - domain_failure MTA options, 39–79
 - domain_uplevel MTA option, 39–81
 - MTA options, 34–29, 39–78
 - Performance tuning, 7–5, 34–30, 39–83, 39–153
 - Rewrite rule, ldap_domain_known_attributes MTA option, 34–30
 - Rewrite rules, \$V and \$Z flags, 34–29
- Forwarding user mail, 35–59
- ims-ms channel, 51–2
- Overview of, 35–3
- Performance tuning, 39–152
 - Domains, 7–5, 34–30, 39–83, 39–153
 - Reverse addresses, 39–153
- Rewrite rules
 - Domain found/not-found in LDAP, 34–29
- Schema
 - Authentication results, 39–151
 - Timeouts on LDAP queries, 39–77
 - User/group lookup MTA options, 39–84
- Directories
 - Archiving
 - store.archive.tmpdir option, 16–14
 - Configuration
 - imta_table, 40–3
 - Dirsync
 - imta_dl, 40–3
 - imta_bin, 40–3
 - imta_dl, 40–3
 - imta_lib, 40–3
 - imta_log, 40–3
 - imta_program, 40–4
 - imta_table, 40–3
 - log
 - imta_log, 40–3
 - Message Store
 - dbtmpdir option, 16–11

- Message Store snapshots
 - snapshotpath Message Store option, 16–9
- MTA options, 39–154
- storedebug under tmpdir
 - autorepairdebug Message Store option, 16–10
- directoryscan SNMP option, 60–2
- disabledestinationfilter channel option, 33–109
- disableetrn channel option, 33–116
- disablesourcefilter channel option, 33–109
- DISABLE_EXPAND TCP/IP-channel-specific option
 - expn* channel options, 33–126
- disconnectbadauthlimit channel option, 33–149
- disconnectbadburllimit channel option, 33–87, 33–120, 49–11
- disconnectbadcommandlimit channel option, 33–87, 33–120
- disconnectcommandlimit channel option, 33–87, 33–120
- disconnectrecipientlimit channel option, 33–87, 33–120
- disconnectrejectlimit channel option, 33–87, 33–120
- disconnecttransactionlimit channel option, 33–124
- Disk quota
 - Domain
 - ldap_domain_attr_disk_quota MTA option, 39–147
 - ldap_domain_attr_message_quota MTA option, 39–148
- Disk space
 - Insufficient for MTA queue
 - error_text_insufficient_disk MTA option, 39–161
 - error_text_insufficient_queue_space MTA option, 39–166
 - Message Store
 - relinker, 16–24
- diskflushinternal Message Store deleted option, 16–26
- diskflushinterval Message Store option
 - Analogous to fsync MTA option, 39–172
- diskusagethreshold Message Store option, 16–5
 - checkdiskusage interaction, 16–5
- Dispatcher, 41–1
 - Access control
 - PORT_ACCESS mapping table, 44–2
 - Autorestart
 - autorestart.enable option, 7–23
 - Debugging
 - debug Dispatcher option, 7–22, 41–3
 - debug_flush MTA option, 39–73, 39–172
 - Operation, 41–1
 - max_conns option, 41–2
 - max_procs option, 41–2
 - min_conns option, 41–2
 - min_procs option, 41–2
 - Worker Processes, 41–2
- Options
 - backlog, 41–3
 - debug, 7–22, 41–3
 - dns_verify_domain, 41–3
 - enable, 41–3
 - Historical interest, 41–11
 - historical_time, 41–4
 - image, 41–4
 - interface_address, See listenaddr, 41–4
 - listenaddr, 41–4
 - logfilename, 41–5
 - max_conns, 41–5
 - max_conns, Operation, 41–2
 - max_handoffs, 41–7
 - max_idle_time, 41–7
 - max_life_time, 41–7
 - max_procs, 41–7
 - max_procs, Operation, 41–2
 - max_shutdown, 41–8
 - min_conns, 41–6
 - min_conns, Operation, 41–2
 - min_procs, 41–8
 - min_procs, Operation, 41–2
 - parameter, 41–8
 - service, listenaddr, 41–5
 - service, max_conns, 41–6
 - service, user, 41–10
 - ssl_ports, 41–9
 - ssl_ports, Compared to maytls* channel options, 33–84, 33–152
 - stacksize, 41–8
 - tcp_ports, 41–9
 - tls_bits_reject_msg, 41–10
 - tls_min_bits, 41–10
 - user, 41–10
 - use_nslog, 27–29, 41–10
- Solaris system parameters, 56–3
- Startup, 39–54
- Virtual memory
 - historical_time option, 41–4
- dispositionchannel channel option, 33–95, 47–22
- DKIM (DomainKeys Identified Mail)
 - Channel options, 33–56
 - dkim* channel options, 33–56
 - dkim_ignore_domains MTA option, 39–154
 - dkim_preserve_domains MTA option, 39–155
 - dkim_remove_domains MTA option, 39–155
 - MAX_PREPEND_INDEX milter spamfilter option, 45–13

- MTA options, 39–154
- dkimignore channel option, 33–56
- dkimpreserve channel option, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_preserve_domains MTA option's effect on, 39–155
- dkimremove channel option, 33–56
 - dkim_ignore_domains MTA option's effect on, 39–154
 - dkim_remove_domains MTA option's effect on, 39–155
- dkim_ignore_domains MTA option, 39–154
 - dkimpreserve interaction, 33–56
 - dkimremove interaction, 33–56
 - dkim_preserve_domains interaction, 39–155
 - dkim_remove_domains interaction, 39–155
- dkim_preserve_domains MTA option, 39–155
 - dkimpreserve interaction, 33–56
- dkim_remove_domains MTA option, 39–155
 - dkimremove interaction, 33–56
- dlopen
 - Rewrite rule routine callouts, 34–25
- dlsym
 - Rewrite rule routine callouts, 34–25
- DMARC
 - Sieve filter to deal with broken DMARC usage, 5–26
- dncomps Base certmap option, 7–23
- dncomps certmap option, 7–23
- DNS lookups
 - A records
 - ipv6out option, 7–18, 27–16
 - ipv6sortorder option, 7–18
 - mailfromdnsverify channel option, 33–129, 33–140
 - MAX_A_RECORDS TCP/IP-channel-specific option, 49–29
 - AAAA records
 - ipv6out option, 7–18, 27–16
 - ipv6sortorder option, 7–18
 - Access control TCP wrappers, 6–1
 - BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 - blocked_mail_from_ips MTA option, 39–155
 - Channel options, 33–137, 33–138
 - CHECK_SOURCE TCP/IP-channel-specific option, 49–22
 - clientlookup submitproxy/vdomain option, 27–9
 - Debugging of
 - TRACE_LEVEL TCP/IP-channel-specific option, 49–36
 - DNS verifications
 - nameservers channel option, 33–138
 - test -rewrite utility, 58–35
 - dnsresolveclient option, 7–16
 - dns_verify callouts, 37–28
 - Domain of envelope From
 - error_text_mailfromdnsverify MTA option, 39–165
 - HOST_NOT_FOUND, returnenvelope channel option, 33–99
 - returnenvelope channel option, 33–99
 - Forward lookups
 - forwardcheck* channel options, 33–139
 - TCP wrapper filters, 6–1
 - Host name
 - Mail routing loops, 52–11
 - IPv6
 - ipv6out option, 7–18, 27–16
 - ipv6sortorder option, 7–18
 - ipv6usegethostbyname option, 7–18
 - Local hostname
 - BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 - MTA options, 39–155
 - MX records
 - AUTH_ACCESS \$M flag, 49–39
 - AUTH_ACCESS \$X flag, 49–39
 - AUTH_ACCESS mapping table consulted before, 49–37
 - lastresort channel option, 33–62, 33–140
 - mailfromdnsverify channel option, 33–129, 33–140
 - MAX_MX_RECORDS TCP/IP-channel-specific option, 49–30
 - Null entry, returnenvelope channel option, 33–99
 - Null entry, return_envelope MTA option, 39–156, 39–216
 - Nameservers, 33–137
 - Null MX record entry
 - error_text_null_mx MTA option, 39–166
 - returnenvelope channel option, 33–99
 - return_envelope MTA option, 39–156, 39–216
 - Reverse lookups
 - BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 - clientlookup submitproxy/vdomain option, 27–9
 - dnsresolveclient option, 7–16
 - ident* channel options, 33–138
 - nameservers channel option, 33–138
 - TCP wrapper filters, 6–1, 6–4
 - SPF, 39–240
 - Debugging, mm_debug MTA option, 39–74

- Permanent errors,
 - spf_smtp_status_permerror MTA option, 39–241
 - spf* channel options, 33–143
 - SPF_LOCAL mapping table avoids actual DNS lookups, 33–146
 - spf_max_dns_queries MTA option, 39–244
 - spf_max_recursion MTA option, 39–244
 - spf_max_time MTA option, 39–244
 - SRS MTA options, 39–244
 - Temporary errors,
 - spf_smtp_status_permerror MTA option, 39–243
- TCP/IP channels, 33–137
- DNS verification services
 - dns_verify_domain Dispatcher option, 41–3
- dnsresolveclient base option, 7–16
- dns_verify lookups
 - blocked_mail_from_ips MTA option, 39–155
- dns_verify.so, 37–28
- dns_verify_domain Dispatcher option, 41–3
- Domain aliases in LDAP
 - Testing of
 - test -rewrite utility, 58–28
- Domain database, 34–34
 - domain_database_url MTA option, 39–202
 - imta_domain_database MTA option (DELETED), 40–8
 - MTA options
 - comment_chars, 34–35
 - Spaces in key or value
 - On-disk crdb format, 34–35
 - TAB character
 - On-disk crdb format, 34–35
 - use_domain_database MTA option, 39–61
- Domain entries in LDAP
 - Testing of
 - test -rewrite utility, 58–28
- Domain map
 - usedomainmap auth option, 12–1
- Domain rewriting rules
 - See Rewrite rules, 34–1
- Domain-based Message Authentication, Reporting & Conformance
 - See DMARC, 5–26
- domainallowed ENS option, 27–13, 61–2
- domainallowed eval_ldapd option, 27–13, 62–1
- domainallowed IMAP option, 21–10, 27–13
- domainallowed IMAP Proxy/POP Proxy/SUBMIT Proxy option, 27–13
- domainallowed MSHTTP option, 22–2, 27–13, 27–13, 29–12
- domainallowed option, 27–13
- domainallowed POP option, 22–2, 27–14
- domainetrn channel option, 33–116
- Domainmap options
 - debug, 7–22
- domainnotallowed ENS option, 27–14, 61–2
- domainnotallowed eval_ldapd option, 27–14, 62–1
- domainnotallowed IMAP option, 21–11, 27–14
- domainnotallowed IMAP Proxy/POP Proxy/SUBMIT Proxy option, 27–13
- domainnotallowed MSHTTP option, 27–14, 29–12
- domainnotallowed option, 27–13
- domainnotallowed POP option, 22–2, 27–14
- Domains
 - Catchall address, 39–86
 - mailDomainCatchallAddress default for ldap_domain_attr_catchall_address, 39–149
 - Catchall mapping
 - mailDomainCatchallMapping default for ldap_domain_attr_catchall_mapping, 39–149
 - Creation date
 - ldap_domain_attr_creation_date MTA option, 39–151
 - Direct LDAP lookups
 - Caching, 34–30
 - Caching, domain_match_cache_size MTA option, 39–152
 - Caching, domain_match_cache_timeout MTA option, 39–152
 - Caching, ldap_domain_timeout MTA option, 39–152
 - Performance tuning, 34–30
 - Vanity, 35–8
 - Forwarding, 35–57
 - IP literal address
 - Rewrite rule handling of, 34–8
 - Spaces in, 34–9
 - LDAP attributes
 - aliasedObjectName default for ldap_domain_attr_alias MTA option, 7–6, 39–143
 - Autoreply timeout semantics, 39–147
 - Autosecretary semantics, 39–146
 - Default mailHost semantics, 39–147
 - Domain disk quota semantics, 39–147
 - Domain message quota semantics, 39–148
 - Domain opt-in to detour routing semantics, 39–150
 - Domain recipient cutoff semantics, 39–150
 - Domain recipient limit semantics, 39–150
 - Domain source channel semantics, 39–150
 - Domain spam/virus package 1 opt-in, 39–146
 - Domain uplevel semantics, 39–143

DomainUidSeparator default for
 ldap_domain_attr_uid_separator MTA
 option, 7–6, 39–144

inetCanonicalDomainName default for
 ldap_domain_attr_canonical MTA option,
 39–143

inetDomainBaseDn default for
 ldap_domain_attr_basedn MTA option, 7–6,
 39–143

inetDomainStatus default for
 ldap_domain_attr_status MTA option, 7–7,
 39–145

ldap_domain_attr_sourceblocklimit MTA
 option, 39–149

mailDomainCatchallAddress default for
 ldap_domain_attr_catchall_address, 39–149

mailDomainCatchallMapping default for
 ldap_domain_attr_catchall_mapping, 39–149

mailDomainConversionTag default for
 ldap_domain_attr_conversion_tag MTA
 option, 39–146

mailDomainMsgMaxBlocks default for
 ldap_domain_attr_blocklimit MTA option,
 39–145

mailDomainReportAddress default for
 ldap_domain_attr_report_address, 39–148

mailDomainSieveRuleSource default for
 ldap_domain_attr_filter, 39–148

mailDomainStatus default for
 ldap_domain_attr_mail_status MTA option,
 7–7, 39–145

mailRoutingHosts, Default for
 ldap_domain_attr_routing_hosts MTA
 option, 39–144

mailRoutingSmartHost default for
 ldap_domain_attr_smarthost MTA option,
 39–144

Nosoliciting semantics, 39–147

objectClass, 39–114

Presence semantics, 39–146

Source conversion tag semantics, 39–146

Spare N attribute, 39–125

Message size limits

- ldap_domain_attr_blocklimit MTA option,
 39–145
- ldap_domain_attr_sourceblocklimit MTA
 option, 39–149

Postmaster

- mailDomainReportAddress default for
 ldap_domain_attr_report_address, 39–148

Routing

- mailRoutingHosts LDAP attribute, 39–144
- mailRoutingSmartHost LDAP attribute,
 39–144
- Trailing dot on name, 34–5
- Vanity, 39–79, 39–86, G–11
 - Direct LDAP lookups, 35–8
 - domain_match_url MTA option, 34–30

domainsearchformat mmp/imaproxy/popproxy/
 vdomain option, 27–14

domainvrfy channel option, 33–125

domain_database_url MTA option, 39–203

domain_failure MTA option

- Direct LDAP domain lookups, 34–30, 34–30

domain_failure MTA options, 39–79

domain_match_cache_size MTA option, 39–152

- Direct LDAP domain lookups, 34–30, 34–30

domain_match_cache_timeout MTA option

- Direct LDAP domain lookups, 34–30, 34–30

domain_match_url MTA option, 39–80

- Direct LDAP domain lookups, 34–30, 34–30
- Example, 35–8

domain_uplevel MTA option, 39–81

- Direct LDAP domain lookups, 34–30, 34–30
- Example, 35–7

dropblank channel option, 33–68

duplicate_timeout_default MTA option, 39–230

E

EAI (Email Address Internationalization), G–4

ehlo channel option, 33–118

ehlokeywords submitproxy option, 27–14

- Analogous to capability IMAP proxy option,
 27–9

ehlokeywords vdomain option, 27–14

eightbit channel option, 33–53, 33–125

eightnegotiate channel option, 33–53, 33–125

eightstrict channel option, 33–53, 33–125

- error_text_unnegotiated_eightbit MTA option,
 39–166

email_body_charset gateway_profile option, 53–5

email_header_charset SMS gateway_profile option,
 53–5

enable autorestart option, 7–23

enable Base autorestart option, 7–23, 7–23

enable Deployment Map option, 14–1

enable Dispatcher option, 41–3

- Default for schedule.task:purge.enable, 41–3

enable ENS option, 61–1

enable IMAP option, 21–3

enable indexer option, 25–1

enable Job Controller option, 42–10

- Default for schedule.task:purge.enable, 42–10
- Default for schedule.task:return_job.enable,
 42–10

enable Message Store dbreplicate option, 16–16
 enable Message Store messagetype option, 16–21
 enable Message Store msghash option, 16–22
 enable Message Store option, 16–4
 enable Message Store purge option, 16–23
 enable Message Store relinker option, 16–24
 enable Message Store typequota option, 16–21, 16–21, 16–22
 enable MeterMaid option, 46–2
 enable MMP option, 27–5
 enable MSHTTP option, 29–3
 enable MTA option, 39–54

- Default for dispatcher.enable, 41–3
- Default for job_controller.enable, 42–10
- Default for schedule.task:purge.enable, 8–3, 16–24
- Default for schedule.task:return_job.enable, 8–3, 8–4, 8–4

enable notifytarget option, 24–2
 enable PAB option, 59–1
 enable POP option, 22–1
 enable rollovermanager option, 15–1
 enable S/MIME option, 30–1
 enable Scheduler option, 8–1
 enable Scheduler task option, 8–2
 enable Scheduler task:msprobe option, 10–1
 enable Scheduler task:return_job option, 8–4, 47–4
 enable Scheduler task:snapshot option, 8–5
 enable Scheduler task:snapshotverify option, 8–5
 enable SMS gateway option, 53–2
 enable SNMP option, 60–1
 enable Store option

- Default for schedule.task:expire.enable, 8–3, 16–18
- Default for schedule.task:snapshot.enable, 8–4, 8–5

enable Watcher option, 9–1
 enablecontextname SNMP option, 60–2
 enablelastaccess base option, 7–9
 enablelog Scheduler option, 8–1
 enablesslport ENS option, 61–1
 enablesslport IMAP option, 21–3
 enablesslport MSHTTP option, 29–3
 enablesslport POP option, 22–2
 enableuserlist IMAP option, 21–11
 enableuserlist MSHTTP option, 29–12
 enable_delay_timers MTA option, 39–70
 enable_sieve_body MTA option, 5–22, 39–228
 enable_sieve_ereject MTA option, 39–228
 enable_sieve_memcache MTA option, 39–228

- Disabling memcache Sieve extension, 5–53

enable_sieve_metermaid MTA option, 39–229

- Disabling metermaid Sieve extension, 5–57

enable_sieve_regex MTA option, 39–229

- regex Sieve extension, 5–61

Encodings

- Message/* and multipart/* parts, 33–46

encryptnew Message Store option, 16–12
 enqueueremoveveroute channel option, 33–37
 ENS, 1

- enshost notifytarget option, 24–2
- ensport notifytarget option, 24–2
- Host
 - base.listenaddr option, 7–17
 - listenaddr base option, 61–1
- Logging
 - Subscribe/unsubscribe events, enssub value in debugkeys option, 27–12
- msprobe probe of, 10–2
- Options, 61–1
 - domainallowed, 27–13, 61–2
 - domainnotallowed, 27–14, 61–2
 - enable, 61–1
 - enablesslport, 61–1
 - Example of settings, 24–1
 - local.store.notifyplugin not used in Unified Configuration, 2–1
 - loglevel, 39–72, 61–2
 - port, 61–1
 - port, Match ensport notifytarget option, 24–2
 - secret, 61–2
 - sslnicknames, 27–27, 61–2
 - sslport, 61–1
- SSL
 - sslport ENS option, 61–1
- Startup, 61–1

enshost notifytarget option, 24–2

- Match base.listenaddr option value, 61–1

ensport notifytarget option, 24–2

- Match ens.port option value, 61–1

ensureownerrights Message Store option, 16–6
 Envelope From address

- Accepted
 - error_text_accepted_return_address MTA option, 39–166
- Adding SMTP AUTH authenticated address, 44–14
- Authenticated sender as, 44–14
- Blank
 - returnenvelope channel option, 33–99
- Domain corresponds to null MX
 - returnenvelope channel option, 33–99
- Empty
 - Distinguishing feature of notification messages, 47–1
- Invalid

- error_text_invalid_return_address MTA option, 39–166
- Mailing list override of
 - ENVELOPE_FROM alias file named parameter, 35–32
 - ldap_errors_to MTA option, 39–137
- Mailing lists
 - alias_envelope_from alias option, 35–14
- Overridden via AUTH_ACCESS mapping, 49–38
- Replace via FROM_ACCESS mapping table, 44–9
- Reply-to: addition
 - ORIGINATOR_REPLY alias file named parameter, 35–37
- Return-path: header field, 33–63
- Sieve filter access to, 5–26
- SMS gateway
 - from_domain SMS gateway_profile option, 53–5
- Unknown
 - error_text_unknown_return_address MTA option, 39–166
 - userswitchchannel effect, 33–82
 - Verifying apparently local addresses are valid
 - returnenvelope channel option, 33–99
 - Verifying it rewrites to an MTA channel
 - returnenvelope channel option, 33–99
 - Verifying its domain resolves in the DNS
 - returnenvelope channel option, 33–99
- Envelope To address
 - Sieve filter access to, 5–26, 36–6
- envelopetunnel channel option, 33–68
- Environment variables
 - APPLICATIONINFO
 - test_smtp_master and test_smtp_slave use of, 52–9
 - CONFIGROOT, 40–2
 - Symbolic names in msconfig option values or recipes, 3–1
 - DATAROOT
 - queuedir msprobe option, 10–1
 - Symbolic names in msconfig option values or recipes, 3–1
 - PMDF_CHANNEL
 - test_smtp_master and test_smtp_slave use of, 52–9
 - SERVERROOT, 40–2, 40–4
 - Base for location of mail.log, 40–7
 - Base for location of mail.log_current, 40–6
 - Base for location of mail.log_yesterday, 40–6
 - Base for location of MTA alias file, 40–6
 - Base for location of MTA compiled charset data, 40–6
 - Base for location of MTA compiled command data, 40–5
 - Base for location of MTA compiled config data, 40–6
 - Base for location of MTA configuration, 40–3
 - Base for location of MTA executables, 40–3, 40–3
 - Base for location of MTA files and configuration, 40–2
 - Base for location of MTA legacy configuration file, 40–5
 - Base for location of MTA log file directory, 40–3
 - Base for location of MTA run-time libraries, 40–3
 - Base for location of MTA Unified Configuration file, 40–5
 - Base for location of option.dat file, 40–5
 - Base for location of pipe channel programs, 40–4
 - Base of location of system Sieve filter file, 40–5
 - Constructing default value for base.tmpdir option, 7–14
 - Message transaction log file location, 40–6
 - Symbolic names in msconfig option values or recipes, 3–1
- TRANSPORTINFO
 - test_smtp_master and test_smtp_slave use of, 52–9
- Errors
 - (bad authentication limit reached; disconnecting), 33–150
 - 4.2.1 cannot reenqueue while still held, 52–11
 - 5.4.6 (SMTP client-server loop detected), 33–128
 - 5.7.1 <Sieve-rejection-text>, 5–29
 - 525 5.7.13 Account disabled, 49–52
 - 530 5.7.0 No AUTH command has been given., 33–150
 - 535 5.7.8 Authorization failure, 49–52
 - 535 5.7.8 Bad username or password, 49–52
 - 550 5.7.1 unknown host or domain
 - Recipient *_ACCESS mapping tables, 44–8
 - 550 5.7.1 you are not allowed to use this address
 - Recipient *_ACCESS mapping tables, 44–8
 - default file
 - Critical level, <service-name> server is not responding, 10–1
 - Error level, unable to connect to <service-name> server:, 10–1

Warning level, <server-name> server took over N seconds to respond!, 10–2

Error: invalid port in tcp_listen, 28–1, 28–1, 41–10

filtering/scanning error, 39–250

IMAP_MAILBOX_EXHAUSTED
maxsearchmailboxes IMAP option, 21–4

IMAP_MAILBOX_LOCKED
Special backoff handling by ims-ms and LMTP client channels, 33–101

Javascript
Webmail clients, charset problems, 29–4

Mailbox is busy
MAILBOX_BUSY_FAST_RETRY TCP/IP-channel-specific option, 49–29

message too sensitive for one or more paths used, 33–107

mm_init
ALIAS_HASH_SIZE exceeds maximum, 39–176
ALIAS_MEMBER_SIZE exceeds maximum, 39–177
authpassword only valid in XML configuration, 33–146
authusername only valid in XML configuration, 33–146
CHANNEL_TABLE_SIZE exceeds maximum, 39–177
DOMAIN_HASH_SIZE exceeds maximum, 39–178
duplicate host in channel table -- ..., 33–79
externalidentify only valid in XML configuration, 33–146
FILE_MEMBER_SIZE exceeds maximum, 39–178
FORWARD_DATA_SIZE exceeds maximum, 39–178
GENERAL_DATA_SIZE exceeds maximum, 39–179
HOST_HASH_SIZE exceeds maximum, 39–179
illegal alias; too long, 35–24
Invalid delivery option clause;, 39–95
LDAP_ATTR_NAME_HASH_SIZE exceeds maximum, 39–179
LDAP_OBJECT_CLASS_HASH_SIZE exceeds maximum, 39–179
MAP_NAMES_SIZE exceeds maximum, 39–180, 39–197
mtprioritiesallowed value must be an integer, 33–106, 33–130
mtprioritiesallowed value outside -9..9 range, 33–106, 33–130
mtprioritiesrequired value must be an integer, 33–106, 33–131
mtprioritiesrequired value outside -9..9 range, 33–106, 33–131
no official host name for channel ..., 33–79
no room in alias member table for alias, 39–177
no room in channel host table for, 39–179
no room in channel table for, 39–177
no room in file member table for file string, 39–178
no room in pool for alias ..., 39–181
no room in pool for charset7 ..., 39–181
no room in pool for charset8 ..., 39–181
no room in pool for conversion data ..., 39–181, 39–181
no room in pool for daemon ..., 39–181
no room in pool for file member string ..., 39–181
no room in pool for forward data, 39–181
no room in pool for general data, 39–181
no room in pool for map entry ..., 39–181
no room in pool for queue ..., 39–181
no room in pool for reverse data, 39–181
no room in pool for value of option ..., 39–181
no room in rewrite rule table for, 39–178
no room in string pool for local alias ..., 39–181
no room in string pool for map entry..., 39–181
No room in table, 58–18
no room in table for alias, 39–176
no room in table for forward data, 39–178
no room in table for general data ..., 39–179
no room in table for mapping named, 39–180, 39–197
no room in table for reverse data, 39–180
official host name is too long -- ..., 33–79
OPTIONS_HASH_SIZE exceeds maximum, 39–180
REVERSE_DATA_SIZE exceeds maximum, 39–180
string pool overflow on pattern - ..., 39–181
Unable to allocate LDAP attribute name hash array., 39–179
Unable to allocate LDAP object class hash array., 39–179
Unable to expand LDAP attribute name hash table, 39–179
Unable to expand LDAP object class hash table, 39–179
MTA options, 39–156
Postmaster mail, 47–1

Returned messages, 47–1

Sieve filter

5.7.1 <Sieve-rejection-text>, 5–29

:content decode not supported for body test, 5–21

:keepmailfrom conflicts with :notify in redirect action, 5–41

:keepmailfrom conflicts with :ret in redirect action, 5–41

:notify conflicts with :keepmailfrom in redirect action, 5–41

:regex only allowed in system-level sieves, enable_sieve_regex MTA option, 39–229

:ret conflicts with :keepmailfrom in redirect action, 5–41

A non system-level sieve script specified warn, 5–63

A non-system level sieve script specified addconversiontag, 5–18, 5–48

A non-system level sieve script specified adjustcounter, 5–18, 5–50

A non-system level sieve script specified removeconversiontag, 5–18, 5–48

A non-system level sieve script specified setconversiontag, 5–18, 5–48

A non-system level sieve script specified setenvelopefrom, 5–8, 5–61

A non-system level sieve script specified transactionlog, 5–18

Adjustcounter value argument must be an integer, 5–50

Argument to addconversiontag must be a string, 5–48

Argument to capture must be a string, 5–51

Argument to envelope must be string or list, 5–27

Argument to removeconversiontag must be a string, 5–48

Argument to setconversiontag must be a string, 5–48

Argument to setenvelopefrom must be a string, 5–8, 5–61

Argument to setmtpriority must be a string or integer, 5–18, 5–62

Argument to setnotify must be a string or list, 5–61

Argument to setoperation must be a string, 5–18, 5–61

Argument to setpriority must be a string or number, 5–20

Argument to setreturn must be a string, 5–61

Argument to warn must be a string, 5–63

Body not listed in require clause prior to use, 5–9, 5–22

Body test only allowed in system-level sieves, 5–22

Cannot build translation map for these charsets: <reason>, 5–62

Cannot combine erefuse with anything but discard, 5–29

Cannot combine ereject with anything but discard, 5–29

Cannot combine reject with anything but discard, 5–29

Capture :message cannot be marked :header, 5–51

Channel argument must be a string, 5–50

Closing "}" is missing from list, 5–3

Comma or closing ")" is missing from <name> routine definition, 5–64

Content argument to body must be string or list, 5–21, 5–46

Content of report message, return_error.txt file, 47–13

Conversion tag result is too long, 5–48

Conversiontag envelope field specified in non-system script, 5–27

Copy not listed in require clause prior to use, 5–22, 5–41

Copy not listed in require clause prior to use, fileinto action, 5–5

Copy not listed in require clause prior to use, redirect action, 5–7

Copy used twice in one action, 5–22

Copy used twice in one fileinto action, 5–5, 5–22

Date not listed in require clause prior to use, 5–23

Date not listed in require clause prior to use, currentdate test, 5–9

Date not listed in require clause prior to use, date test, 5–9

Destination charset name must be a string, 5–62

Dsn-redirect not listed in require clause prior to use, 5–41

Duplicate not listed in require clause prior to use, 5–9, 5–24

Duplicate test is not available, Seen when duplicate_tracking_url is not set (length 0), 5–24

Editheader not listed in require clause prior to deleteheader use, 5–4, 5–25

Editheader not listed in require clause prior to replaceheader use, 5–7, 5–25

editheader not listed in require clause prior to use of deleteheader/addheader action, 5-4, 5-25
 Enotify not listed in require clause prior to notify use, 5-16
 Enotify not listed in require clause prior to notify_method_capability use, 5-12
 Enotify not listed in require clause prior to use, 5-16
 Envelope not listed in require clause prior to use, 5-10, 5-26
 Envelope-auth not listed in require clause prior to use, 5-14, 5-26
 Envelope-dsn not listed in require clause prior to use, 5-26
 Envelope-dsn not listed in require clause prior to use, envid, 5-15
 Envelope-dsn not listed in require clause prior to use, notify, 5-15
 Envelope-dsn not listed in require clause prior to use, orcpt, 5-15
 Envelope-dsn not listed in require clause prior to use, ret, 5-15
 Environment not listed in require clause prior to use, 5-15
 Ereject not listed in require clause prior to use, 5-4, 5-28
 Ereject not listed in require clause prior to use, Issued when enable_sieve_ereject=0, 39-228
 Error decoding zone argument to currentdate test, 5-24
 Error decoding zone argument to date, 5-24
 Error in sieve filter: List too large, max_sieve_list_size MTA option, 39-227
 Error in sieve filter: Resultant string is too long, 39-227
 Extlist not listed in require clause prior to use, 5-16
 Extlists not listed in require clause prior to use, 5-16, 5-26
 Extlists not listed in require clause prior to valid_ext_list use, 5-12
 Extracttext not listed in require clause prior to use, 5-4, 5-38
 Fileinto cannot be combined with jettison, 5-22
 Fileinto cannot be combined with refuse, 5-29
 Fileinto cannot be combined with reject, 5-29
 Fileinto not allowed in this type of filter, 5-5, 5-37
 Fileinto not listed in require clause prior to use, 5-5, 5-36
 First addheader string too long for header label, 5-25
 Foreverypart not listed in require clause prior to use, 5-4, 5-38
 Ihave not listed in require clause prior to use, 5-11, 5-37
 Illegal terminal element found in expression, 5-3
 Imap4flags not listed in require clause prior to addflag use, 5-3, 5-37
 Imap4flags not listed in require clause prior to hasflag use, 5-10, 5-37
 Imap4flags not listed in require clause prior to removeflag use, 5-7, 5-37
 Imap4flags not listed in require clause prior to setflag use, 5-8, 5-37
 Imap4flags not listed in require clause prior to use, 5-37
 Imap4flags not listed in require clause prior to use of flag action, 5-3, 5-7, 5-8, 5-10, 5-37
 Imap4flags not listed in require clause prior to use, fileinto :flags, 5-5
 Imap4flags not listed in require clause prior to use, keep :flags, 5-5
 Improperly terminated {} block, 5-3
 Improperly terminated {} structure, 5-64
 Index not listed in require clause prior to use, 5-15, 5-23
 Invalid argument <argument-string> to setoperation, 5-18, 5-61
 Invalid argument <argument-string> to setreturn, 5-61
 Invalid destination charset name <string>, 5-62
 Invalid header field name in addheader, 5-25
 Invalid routine parameter list, 5-64
 Invalid source charset name <string>, 5-62
 Jettison cannot be combined with anything but discard, 5-22
 Jettison not listed in require clause prior to use, 5-18, 5-22
 Keep cannot be combined with jettison, 5-22
 Keep cannot be combined with refuse, 5-29
 Keep cannot be combined with reject, 5-29
 Left hand side of assignment must be a variable, 5-2
 Loop is disabled, 5-52
 Maximum number of duplicate tests exceeded, 5-24
 Maximum number of duplicate tests exceeded, max_duplicates MTA option, 39-226, 39-231
 Memcache access has been disabled, 5-53

Memcache access has been disabled, enable_sieve_memcache MTA option, 39–229

Memcache only allowed in system-level sieves, 5–53

Memcache only allowed in system-level sieves, enable_sieve_memcache MTA option, 39–229

Metermaid access has been disabled, 5–57

MeterMaid access has been disabled, enable_sieve_metermaid MTA option, 39–229

Metermaid only allowed in system-level sieves, 5–57

Metermaid only allowed in system-level sieves, enable_sieve_metermaid MTA option, 39–229

Mime not listed in require clause prior to use, 5–16, 5–38

Multiple :duplicate arguments given to adjustcounter, 5–50

Multiple :zone:/originalzone arguments given to date, 5–24

Multiple channel arguments given to adjustcounter, 5–50

Multiple delay arguments to setnotify, 5–61

Multiple failure arguments to setnotify, 5–61

Multiple header arguments given to capture, 5–51

Multiple success arguments to setnotify, 5–61

Multiple type arguments given to capture, 5–51

Multiple zone arguments given to currentdate test, 5–24

Name too long for variable: <variable-name>, 5–47

Nested routine definitions not allowed, 5–64

Never combined with delay in setnotify, 5–61

Never combined with failure in setnotify, 5–61

Never combined with success in setnotify, 5–61

No room in table for variable: <variable-name>, 5–47

No room in table for variable:, max_variables MTA option, 39–227

Nonotify specified in a non-system level sieve script, 5–18, 5–40

Notify :from argument is not a valid address, 5–40, 39–225

Notify mailto: recipient is not a valid address, 5–40, 39–225

Notify/enotify not listed in require clause prior to notify use, 5–16

Notify/enotify not listed in require clause prior to use, 5–16, 5–16

Novacation specified in a non-system level sieve script, 5–18, 5–44

Only :contains and :is matches supported for body test, 5–21, 5–61

Override not listed in require clause prior to use, 5–18, 5–67, 5–67

Parameter <name> already declared, 5–64

Redirect cannot be combined with jettison, 5–22, 5–40

Redirect cannot be combined with refuse, 5–29, 5–40

Redirect cannot be combined with reject, 5–29, 5–40

Refuse not listed in require clause prior to use, 5–7, 5–28

Refuse specified in a non-system sieve script, MS 6.1, 5–28

Regex not listed in require clause prior to use, 5–18, 5–60

Reject not listed in require clause prior to use, 5–7, 5–28

Relational not listed in require clause prior to use, 5–26, 5–41

Relational not listed in require clause prior to use with <test-name>, 5–41

Relational not listed in require clause prior to use, :count, 5–15

Relational not listed in require clause prior to use, :value, 5–16

Resultant string is too long, 39–227

Return can only be used inside a routine, 5–64

Routine <name> <n> parameters exceeds 64 maximum, 5–64

Setmtpriority specified in a non-system level sieve script, 5–18, 5–62

Setnotify "NEVER" argument cannot be combined with other values, 5–61

Setnotify specified in a non-system level sieve script, 5–18, 5–61

Setoperation specified in a non-system level sieve script, 5–18, 5–61

Setpriority specified in a non-system level sieve script, 5–18, 5–62

Setreturn specified in a non-system level sieve script, 5–18, 5–61

Sieve counter index must be an integer, 5–50

Sieve counter index out of range, 5–50

Sieve variables have been disabled, max_variables MTA option, 5–47

Sieve variables have been disabled; <test-action> not possible, 5–16

Source charset name cannot be blank, 5-62

Source charset name must be a string, 5-62

Spamtest not listed in require clause prior to use, 5-12, 5-42

Spamtestplus not listed in require clause prior to use, 5-12, 5-42

Strongrandom specified in a non-system level sieve script, 5-18

Sub has been disabled, 5-65

Sub not followed by a valid unused routine name, 5-64

Sub not followed by routine definition, 5-64

Subaddress not listed in require clause prior to use, 5-21, 5-26, 5-43

Subaddress not listed in require clause prior to use, :detail, 5-13

Subaddress not listed in require clause prior to use, :user, 5-14

Too few arguments given to adjustcounter, 5-50

Too few arguments given to capture, 5-51

Too many addheaders specified in user sieve, max_addheaders MTA option, 5-25, 39-225

Too many addheaders specified, max_addheaders MTA option, 39-225

Too many addheaders specified, max_addheaders MTA option MS 7.0.5 and earlier, 5-25

Too many arguments given to adjustcounter, 5-50

Too many arguments given to capture, 5-51

Too many fileintos specified in user sieve, 5-5, 5-37

Too many fileintos specified, max_fileintos MTA option, 39-226

Too many notifys specified, 5-40

Too many notifys specified, max_notifys MTA option, 39-226

Too many redirects specified, 5-40

Too many redirects specified, max_redirects MTA option, 39-226

Too many vacations specified, max_vacations MTA option, 5-44, 5-45, 39-227

Unable to open memcache connection for vacation: <reason>, 5-45

Undefined function or variable "<name>" referenced, 5-2

Unknown channel value '<name>' given to adjustcounter, 5-50

Unknown comparator <comparator-name> given to <test-name>, 5-51

Unknown function required: <name>, 5-2

Unknown namespace <name> specified, 5-47

Unknown namespace specified in variable <name>, 5-47

Unknown or illegal tagged argument: <tag-name>, 5-3

Unknown tag <string> given to adjustcounter, 5-50

Unknown tag <string> given to capture, 5-51

Unrecognized setnotify argument <string>, 5-61

Vacation file <file-name> cannot be opened, 5-45

Vacation not listed in require clause prior to use, 5-8, 5-43, 5-44, 39-227

Vacation specified in a system level sieve script, 5-8, 5-43, 5-46

Vacation-seconds not listed in require clause prior to use, 5-9, 5-43, 5-44, 39-227

Value argument to importanceadjust must be a string, 5-52

Variables not listed in require clause prior to string test use, 5-12, 5-46

Variables not listed in require clause prior to use, 5-46

Variables not listed in require clause prior to use in <test-action-name>, 5-16, 5-46

Variables not listed in require clause prior to use of <test-action-name> action, 5-16, 5-46

Variables not listed in require clause prior to use, set, 5-7

Variables not listed in require clause prior to use, string, 5-12

Virustest not listed in require clause prior to use, 5-12, 5-42

Zero :days and :noaddresses given to vacation, 5-44

Zero :hours and :noaddresses given to vacation, 5-44

Zero :seconds and :noaddresses given to vacation, 5-44

Zone argument to currentdate test must be a string, 5-24

Zone argument to date must be a string, 5-24

{ } block not allowed in this context, 5-3

SMTP client-server loop detected, 33-128

Too many mailboxes

 maxsearchmailboxes IMAP option, 21-4

error_text_* MTA options, 39-157

error_text_accepted_return_address MTA option, 39-166

error_text_access_failure MTA option, 39-158

error_text_alias_auth MTA option, 39-158

error_text_alias_fileerror MTA option, 39-158

- Effect of use_permanent_error MTA option, 39–169
- error_text_alias_fileexist MTA option, 39–158
 - Effect of use_permanent_error MTA option, 39–169
- error_text_alias_locked MTA option, 39–158
- error_text_alias_temp MTA option, 39–158
- error_text_block_over MTA option, 39–159
- error_text_deleted_group MTA option, 39–162
- error_text_deleted_user MTA option, 39–162
- error_text_disabled_alias MTA option, 39–161
 - Effect of use_temporary_error MTA option, 39–169
- error_text_disabled_group MTA option, 39–162
- error_text_disabled_user MTA option, 39–161
 - Effect of use_temporary_error MTA option, 39–169
- error_text_inactive_group MTA option
 - Effect of use_permanent_error MTA option, 39–169
- error_text_inactive_user MTA option
 - Effect of use_permanent_error MTA option, 39–169
- error_text_over_quota MTA option
 - Effect of use_permanent_error MTA option, 39–169
- error_text_recipient_over MTA option
 - Effect of use_permanent_error MTA option, 39–169
 - recipientlimit channel option, 33–87, 33–121
- error_text_spf_* MTA options
 - spfmailfrom and spfrcptto channel options, 33–145
- error_text_still_held MTA option
 - Hold channel, 52–11
- error_text_transaction_limit_exceeded MTA option
 - transactionlimit channel option, 33–125
- error_text_unknown_alias MTA option
 - Effect of use_temporary_error MTA option, 39–169
- error_text_unknown_host MTA option
 - Effect of use_temporary_error MTA option, 39–169
- error_text_unknown_user MTA option
 - Effect of use_temporary_error MTA option, 39–170
- error_text_wrong_account MTA option, 39–161
 - checkrrvs channel option, 33–36, 33–119
- error_text_wrong_domain MTA option, 39–161
 - checkrrvs channel option, 33–36, 33–119
- errsendpost channel option, 33–94, 47–1
- errwarnpost channel option, 33–94, 47–1
- esme_address_npi SMPP server option, 53–11
- esme_address_range SMPP server option, 53–11
- esme_address_ton SMPP server option, 53–12
- esme_password SMPP server option, 53–12
- esme_system_id SMPP server option, 53–12
- esme_system_type SMPP server option, 53–12
- eval_ldapd
 - Options
 - domainallowed, 27–13, 62–1
 - domainnotallowed, 27–13, 62–1
- eval_ldapd options, 62–1
- Event log (NT)
 - Notices generated by address access mapping tables, 44–9
- Exclamation point
 - Alias file
 - Comment line, 35–25
 - Comment line in MTA configuration files
 - comment_chars MTA option, 39–171
- exclusive Message Store expirerule option, 16–19
- expandable_default MTA option, 39–184
- expandchannel channel option, 33–58, 33–89, 33–103, 33–113
- expandlimit channel option, 33–58, 33–89, 33–103, 33–113, 52–18
- expirerule Message Store option, 16–13
- expirstart Message Store deleted option, 16–26
- expirysource channel option, 33–68, 33–105
- expirytime logfile option, 7–19
- explicitaslexternal channel option, 33–151
- exploglevel Message Store expire option, 16–18
- expnallow channel option, 33–126
- expndefault channel option, 33–126
- expndisable channel option, 33–126
- exproute channel option, 33–38
 - exproute_forward MTA option, 39–57
- exproute_forward MTA option
 - exproute channel option, 33–38
- expungemsg notifytarget option, 24–6
- expungesynclevel Message Store option, 16–6
- External filtering context MTA options, 39–170
- externalidentity channel option, 33–146
- extldap: URLs
 - Example, 36–15
 - MTA URL types, 1–4
- extldaps: URLs
 - MTA URL types, 1–4
- extrauseridpattr MSHTTP option, 29–8
- extra_capabilities IMAP option, 21–7

F

- failovertimeout submitproxy option, 27–15
- failovertimeout vdomain option, 27–15
- fdirectory MTA option, 39–172

file: URLs
 Example in memberURL attribute's value, 36–10
 MTA URL types, 1–4

fileinto channel option, 33–110
 ims-ms channels, 33–110, 51–1
 LMTP client (tcp_lmtpcs*) channels, 33–111
 Subaddresses in addresses, 33–110

FILEINTO ims-ms-channel-specific option
 fileinto channel option, 33–111
 Subaddresses in addresses, 35–45

filemode logfile option, 7–19

Files
 \$SERVERROOT/data/queue/tcp_*/spool/*.data-failed, 49–33
 *.data-failed, 49–26, 49–33
 F modifier in MTA connection transaction log entry, 55–9

Access to
 umask Message Store option, 16–13

aliasesdb, 40–7

alias_file
 imta_alias_file, 40–6

charsets.txt, 38–18, 53–5
 Character set names, 33–52

charset_data
 chbuild utility, 58–9
 imta_charset_data, 40–6

check_memcache.so, 37–24

check_metermaid.so, 37–27

command_data
 imta_command_data, 40–5

config.xml, 1
 imta_xml_config_file, 40–5

config_data
 imta_config_data, 40–6

connection.log, 40–7

connection.log*
 separate_connection_log MTA option, 39–270

connection.log_current, 40–7

connection.log_yesterday, 40–7

conversions, 38–1, 40–9
 Legacy configuration, 39–69

countries.txt
 ldap_preferred_country MTA option, 39–119

Creation of
 filemode logfile option, 7–19
 osync MTA option, 39–174
 umask Message Store option, 16–13

daily_cleanup, 39–271

disposition_*.txt, 47–18

disposition_deleted.txt
 Example, 47–20

disposition_dispatched.txt
 Example, 47–20

disposition_option.opt, 47–18, 47–20

disposition_prefix.txt
 Example, 47–20

disposition_suffix.txt
 Example, 47–20

dns_verify.so, 37–28

domaindb, 40–8

Format
 MTA options, 39–170

forward.txt, 40–7

forwarddb, 40–8

general.txt, 40–7

generaldb, 40–8

hold_list, 40–9

imapcmd
 logcommands IMAP option, 21–4

imta
 Debugging, 33–86
 ims-ms channel debugging, 51–7, 51–7
 loglevel MTA option, 39–72, 49–16

imta.cnf
 imta_config_file, 40–5

imta.filter, 40–5
 -system_filter switch of test -rewrite utility, 58–38
 Legacy configuration, 39–223
 Sieve hierarchy, 5–65

imta_alias_file, 40–6

imta_charset_data, 40–6

imta_command_data, 40–5

imta_config_data, 40–6

imta_config_file, 40–5

imta_primary_log, 40–6

IMTA_ROOT:data/sms_gateway_cache, 53–3

imta_secondary_log, 40–6

imta_tertiary_log, 40–7

imta_xml_config_file, 40–5

internet.rules, 34–10
 error_text_unknown_host MTA option, 39–158
 New rewrite rule consults tlds.txt, 34–13
 Used in past instead of . match-all rewrite rule, 34–13

job_controller.cnf, 42–10

job_controller.log-*, 58–8
 debug Job Controller option, 7–22, 42–10

libimtamap.so, 40–9

libimtautil.so, 40–9

Log files
 Access to, umask Message Store option, 16–13

mail.log
 imta_tertiary_log, 40–7

- mail.log*
 - separate_connection_log MTA option, 39–270
- mail.log_current
 - imta_primary_log, 40–6
- mail.log_yesterday
 - imta_secondary_log, 40–6
- mappings, 40–9
 - Legacy configuration, 38–2
- mappings.locale, 47–21
 - DISPOSITION_LANGUAGE mapping table, 47–17
- maximum.dat
 - cnbuild utility, 58–15
- maximum_charset.dat
 - chbuild utility, 58–10
- msgtrace, 23–1, 33–86
 - Debugging, 33–86
 - ims-ms channel debugging, 51–6, 51–7
- mtasdkhdr.h, 33–156
- name_content.dat, 40–7
- nsswitch.conf, 33–137
- option.dat
 - delivery_options MTA option, 39–95
 - imta_option_file MTA Tailor option, 40–5
- option_charset.dat, 40–9
 - chbuild utility, 58–10
- pmdf.cld
 - imta_command_data MTA tailor option, 40–6
- pmdf_err.h, 38–15
- restricted.cnf, 1, 40–10
 - group, imta_world_group old Tailor option, 40–10
 - noprivuser, imta_user_username MTA tailor option, 40–10
 - pipeuser option, 33–62, 33–107
 - user option, 7–11
 - user option, Defragment database sharing over NFS, 52–6
 - user option, Vacation response file sharing over NFS, 39–67
 - user, imta_user MTA tailor option, 40–10
- return templates
 - notary_quote MTA option, 39–174, 39–214
- return_*.*, 47–9
- return_*.txt, 47–10
- return_bounced.txt
 - Example, 47–12
- return_capture.txt
 - Example, 47–12
- return_deferred.txt
 - Example, 47–12
- return_delayed.txt
 - Example, 47–12
- return_delivered.txt
 - Example, 47–13
- return_error.txt
 - Example, 47–13
 - Sieve syntax error, 39–227, 39–227
- return_failed.txt
 - Example, 47–13
- return_forwarded.txt
 - Example, 47–13
- return_header.opt, 47–10
 - Example, 47–10
 - MDN generation, 47–18
- return_option.opt, 47–13
 - MDN generation, 47–18
- return_prefix.txt, 47–12
- return_suffix.txt
 - Example, 47–13
- return_timedout.txt
 - Example, 47–13
- reverse.txt, 40–7
- reversedb, 40–8
- security.cnf, 40–10
- sslpassword.conf, 13–1
- ssrdb, 40–8
- sysexits.h
 - Pipe channel, 52–13
- test_smtp_master, 52–9
- test_smtp_slave, 52–9
- tlds.txt, 34–10, 34–32
 - Consulted by rewrite rule in internet.rules, 34–13
 - error_text_unknown_host MTA option, 39–158
 - Fetching from IANA, 34–32
 - Updating, chbuild utility, 58–9
 - Use chbuild rather than cnbuild, 58–15
- umask Message Store option, 16–13
- vdmap.cfg, 27–30
- Writing of
 - fsync MTA option, 39–172
- filesperjob channel option, 33–99
- file_member_size MTA option, 39–178
- filter channel option, 33–109
 - Sieve hierarchy, 5–65
- filtercomps Base certmap option, 7–23
- filtercomps certmap option, 7–23
- filterhiddenmailinglists MSHTTP option, 29–4
- filterurl base option, 7–8
- filter_cache_size MTA option, 39–228
- filter_cache_timeout MTA option, 39–228
- filter_debug MTA option, 39–74, 39–231
- filter_discard channel, 52–7
 - log_username field in transaction entries, 39–269

- Retrieving messages from, 52–8
 - finalcheckpoint Message Store option, 16–6
 - firstwarn pwexpirealert option, 21–13
 - fixinternaldate IMAP option, 21–3
 - fixsyntaxerrors channel option, 33–68
 - filename Message Store message type mtindex option, 16–22
 - flagtransfer channel option, 33–108, 33–123
 - Previously discarded message flag, 52–9
 - flushinterval logfile option, 7–19
 - folderlockcount Message Store option, 16–11
 - folderpattern Message Store expirerule option, 16–19
 - Folders
 - Delivery to
 - fileinto channel option, 33–110
 - FILEINTO ims-ms-channel-specific option, 51–6
 - FILEINTO ims-ms-channel-specific option, Interaction with fileinto channel option, 33–111
 - Sieve filter fileinto, 5–1
 - Subaddress, 33–42
 - Subaddress, deliveryflags channel option, 33–108, 33–123
 - Subaddresses, In aliases, 35–44
 - Expiring messages from
 - folderpattern expirerule option, 16–19
 - foldersizebytes expirerule option, 16–19
 - folderlockcount Message Store option, 16–11
 - Hidden
 - ensureownerrights Message Store option, 16–6
 - maxfolders Message Store option, 16–6
 - Maximum messages per
 - maxmessages Message Store option, 16–7
 - Pinned
 - pin Message Store option, 16–7
 - privatesharedfolders Message Store options, 16–25
 - publicsharedfolders Message Store options, 16–25
 - Shared
 - listimplicit Message Store option, 16–6
 - Private, restrictanyone Message Store privatesharedfolders option, 16–25
 - Private, restrictdomain Message Store privatesharedfolders option, 16–25
 - publicsharedfolders Message Store option, 16–25
 - sharedfolders Message Store option, 16–8
 - shareflags Message Store privatesharedfolders option, 16–25
 - Undeletable
 - pin Message Store option, 16–7
 - foldersizebytes Message Store expirerule option, 16–19
 - folderurl base option, 7–8
 - forcenbsptospace MSHTTP option, 29–4
 - forcetelemetry icapservice option, 32–1
 - forcetelemetry IMAP option, 21–11
 - forcetelemetry MSHTTP option, 29–12
 - forcetelemetry POP option, 22–3
 - foreground sms_gateway option, 53–3
 - form_names MTA option, 39–272
 - Forward database, 35–61
 - comment_chars MTA option, 39–171
 - Consulted after FORWARD mapping table, 35–60
 - forward_database_url MTA option, 39–203
 - forward_data_size MTA option, 39–178
 - MTA options
 - comment_chars, 39–171
 - forward_data_size, 39–178
 - string_pool_size_3, 39–181
 - use_text_databases, 39–175
 - Source specific entries, 35–61
 - string_pool_size_3 MTA option, 39–181
 - use_text_databases MTA option, 39–175
 - forwardcheckdelete channel option, 33–138
 - forwardchecknone channel option, 33–138
 - forwardchecktag channel option, 33–138
 - Forwarding
 - Testing of
 - test -rewrite utility, 58–28
 - forward_database_url MTA option, 39–203
 - forward_data_size MTA option, 39–178
 - from_domain gateway_profile option, 53–5
 - fsync MTA option, 39–172
 - fullfromheader MSHTTP option, 29–8
 - futurerelease channel option, 33–104, 33–126
- ## G
- General database, 37–20
 - Backslash quoting
 - In-memory format, 37–20
 - Callout from mapping tables
 - Example, 37–19
 - Performance, 37–18
 - Callout from rewrite rules, 34–24
 - Comment lines
 - In-memory format, 37–21
 - comment_chars MTA option, 39–171
 - Examples
 - Sieve external list, 5–34
 - File protection of

- Rewrite rule callout, 34–25
- general_database_url MTA option, 39–203
- general_data_size MTA option, 39–178
- Mapping table lookups, 37–16
- MTA options
 - comment_chars, 37–21, 39–171
 - general_data_size, 37–21, 39–178
 - string_pool_size_3, 37–21, 39–181
 - use_text_databases, 39–175
- Rewrite rule substitution, 34–24
- Spaces in key or value
 - In-memory format, 37–20
 - On-disk crdb format, 37–21
- string_pool_size_3 MTA option, 39–181
- TAB character
 - Converted to space, In-memory format, 37–20
 - On-disk crdb format, 37–21
- use_text_databases MTA option, 39–175
- general_database_url MTA option, 39–203
- general_data_size MTA option, 39–178
 - General database, 37–21
- generatemessagehash channel option, 33–90
 - vnd.oracle.message-hash Sieve environment item, 5–15
- generatereceivedheader MSHTTP option, 29–5
- Generic SMTP channels, 52–9
- Grey-listing
 - Sieve metermaid :greylisting operation, 5–59
- group_dn_template MTA option, 39–96
 - Mailing list membership, 36–10
- gzipattach MSHTTP option, 29–6
- gzippedynamic MSHTTP option, 29–6
- gzipstatic MSHTTP option, 29–6

H

- has_plain_passwords auth option, 12–1
- Header
 - A1-Format:
 - IN-A1-FORMAT conversion entry parameter, 38–9
 - OUT-A1-FORMAT conversion entry parameter, 38–10
 - A1-Type:
 - IN-A1-TYPE conversion entry parameter, 38–9
 - OUT-A1-TYPE conversion entry parameter, 38–10
 - Accept-language:
 - DISPOSITION_LANGUAGE mapping table probe, 47–17
 - language channel option, 33–72, 33–96
 - ldap_spare_4, ldap_spare_5, ldap_spare_6 values, 39–125

- NOTIFICATION_LANGUAGE mapping table probe, 47–9
- Vacation message, Choice of body text, 39–127, 39–128
- Vacation message, Choice of Subject:, 39–127
- Adding of
 - ADD header trimming option, 33–156
 - addheader Sieve action, 5–25
 - FILL header trimming option, 33–156
 - ldap_add_header MTA option, 39–139
 - spamfilter*_includeheaders MTA option, 39–238
 - spamfilter*_received MTA option, 39–239
 - spamfilter*_returnpath MTA option, 39–239
 - Via address access mapping tables, 44–9
- alias_header_addition alias option, 35–15
- alias_header_trim alias option, 35–15
- Approved:
 - alias_password alias option, 35–20
 - ldap_auth_password MTA option, 39–133
 - Mass mailings, 36–20
 - mgrpBroadcasterPolicy LDAP attribute, 39–108
 - Moderated mailing list example, 36–5
 - PASSWORD alias file named parameter, 35–37
 - Password-protected mailing lists, 36–3
- Authentication-results:
 - Sieve filter test on, 5–51
- Auto-submitted:
 - deleteheader Sieve action cannot delete, 5–4
 - Values other than "no" disables sending back a vacation message, 5–45
- Auto-Submitted:
 - MDNs, 47–18
- Bcc:
 - Blank, missingrecipientpolicy channel option, 33–39, 33–74
 - Blank, missing_recipient_policy MTA option, 39–59
 - Generated when original message had no recipient header lines, 39–58
 - message_hash_fields MTA option, 54–19
 - passyntaxerrors channel option, 33–68
 - Use in Axs:One archiving,
 - userheaderecipients Message Store archive option, 16–15
- Buffer overruns
 - Content-disposition: parameters, 33–49
 - Content-type: parameters, 33–49
- Cc:
 - dropblank channel option, 33–68
 - message_hash_fields MTA option, 54–19

-
- passyntaxerrors channel option, 33–68
 - Use in Axs:One archiving, userheaderecipients Message Store archive option, 16–15
 - Client-ip:, 57–3
 - Comment strings, 33–65, 35–55
 - mail_off MTA option, 39–185
 - Content-annotation:
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-comments:
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-description:
 - IN-DESCRIPTION conversion entry parameter, 38–9
 - IN-DISPOSITION conversion entry parameter, 38–9
 - message_hash_fields MTA option, 54–19
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-Description:
 - OUT-DESCRIPTION conversion entry parameter, 38–10
 - Content-disposition:
 - Buffer overruns, 33–49
 - FILENAME parameter length limit, 33–49
 - msexchange channel option, 33–49, 33–130, 33–152
 - nameparameterlengthlimit channel option, 33–49
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Parameter length limits, 33–49
 - Parameter length limits, RFC 2231
 - segmentation of long values, 33–50, 33–54
 - parameterlengthlimit channel option, 33–49
 - Content-Disposition:
 - OUT-DISPOSITION conversion entry parameter, 38–10
 - Parameters, DPARAMETER-COPY-n conversion entry parameters, 38–10
 - Parameters, DPARAMETER-SYMBOL-n conversion entry parameters, 38–10
 - Parameters, IN-DPARAMETER-DEFAULT-n conversion entry parameters, 38–9
 - Parameters, IN-DPARAMETER-NAME-n conversion entry parameters, 38–9
 - Parameters, IN-DPARAMETER-VALUE-n conversion entry parameters, 38–9
 - Parameters, OUT-DPARAMETER-NAME-n conversion entry parameters, 38–11
 - Parameters, OUT-DPARAMETER-VALUE-n conversion entry parameters, 38–11
 - Content-id:
 - message_hash_fields MTA option, 54–19
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-language:
 - CHARSET-CONVERSION mapping table, 38–19
 - IN-LANGUAGE conversion entry parameter, 38–9
 - OUT-LANGUAGE conversion entry parameter, 38–11
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - return_prefix.txt, 47–12
 - See RFC 3066 (Tags for the Identification of Languages), 47–9
 - Content-mode:
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-transfer-encoding:
 - ignoremessageencoding channel option, 33–46
 - ignoremultipartencoding channel option, 33–46
 - IN-ENCODING conversion entry parameter, 38–9
 - interpretmessageencoding channel option, 33–46
 - interpretmultipartencoding channel option, 33–46
 - OUT-ENCODING conversion entry parameter, 38–11
 - OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 - Content-type:
 - application/batch-smtp, 50–3
 - Buffer overruns, 33–49
 - Charset labelling forces addition of, 33–52
 - charset parameter, 33–52, 38–16
 - IN-SUBTYPE conversion entry parameter, 38–9
 - IN-TYPE conversion entry parameter, 38–9
 - Inspected by conversion channel, 38–7
 - Message Store message typing, header
 - Message Store messagetype option, 16–20, 16–21
 - message/partial, 33–48
 - message/partial, defragment channel option, 33–46
 - message/partial, Defragmentation channel, 52–3

message_hash_fields MTA option, 54–19
 multipart/report, DSNs, 47–1
 multipart/report, MDNs, 47–1
 NAME parameter length limit, 33–49
 nameparameterlengthlimit channel option, 33–49
 OUT-SUBTYPE conversion entry parameter, 38–11
 OUT-TYPE conversion entry parameter, 38–11
 OVERRIDE-HEADER-FILE conversion entry parameter, 38–15
 Parameter length limits, 33–49
 Parameter length limits, RFC 2231
 segmentation of long values, 33–50, 33–54
 parameterlengthlimit channel option, 33–49
 Parameters, OUT-PARAMETER-NAME-n conversion entry parameters, 38–11
 Parameters, OUT-PARAMETER-VALUE-n conversion entry parameters, 38–11
 Parameters, PARAMETER-COPY-n conversion entry parameters, 38–11
 Parameters, PARAMETER-SYMBOL-* conversion entry parameters, 38–10
 text/calendar, msexchange channel option, 33–49, 33–130, 33–152
 Content-Type:
 Parameters, IN-PARAMETER-DEFAULT-n conversion entry parameters, 38–9
 Parameters, IN-PARAMETER-NAME-n conversion entry parameters, 38–9
 Parameters, IN-PARAMETER-VALUE-n conversion entry parameters, 38–9
 Date-warning:, 33–120
 Not generated with SMTP SUBMIT, 33–119
 received_domain MTA option, 39–222
 Date:
 Absence ignored in passthrough mode, 33–120
 Axs:One PostedDate, 16–15
 Day of the week, 33–66
 DSN human-readable part, 47–10
 Format of date, 33–66
 Not required with SMTP SUBMIT, 33–119
 Required in relay mode, 33–120
 Tests on, See also Sieve filters, date extension, 5–23
 Used for date archiving, 45–9
 usesentdate MSHTTP option, 29–6
 Deferred-delivery:, 33–102, 33–102
 alias_deferred alias option, 35–12
 alias_deferred_list alias option, 35–12
 alias_deferred_mapping alias option, 35–12
 Compared to FUTURERELEASE, 49–12
 Compared to FUTURERELEASE SMTP extension, 33–103
 Compared with futurerelease, 33–104, 33–127
 Compared with futurerelease functionality, 33–104, 33–126
 DEFERRED named alias file parameter, 35–31
 DEFERRED_LIST named alias file parameter, 35–31
 DEFERRED_MAPPING named alias file parameter, 35–31
 Defragment-failed:, 52–5
 Detecting end of, 33–73
 Disposition-notification-to:, 47–1
 DKIM-Signature:
 destinationdkim* channel options, 33–56
 dkim* channel options, 33–56
 dkim_ignore_domains MTA option, 39–154
 dkim_preserve_domains MTA option, 39–155, 39–155
 dkim_remove_domains MTA option, 39–155, 39–155
 MTA options, 39–154
 Encoding:
 ignoreencoding channel option, 33–46
 interpretencoding channel option, 33–46
 envelopetunnel channel option, 33–68
 Errors-to:
 error-return-address alias file positional parameter, 35–14
 Expires:
 Defined in RFC 2156 (MIXER) to replace Expiry-date:, 33–68, 33–105
 Expiry-date:
 alias_expiry alias option, 35–15
 Defined in RFC 1327 (Mapping between X.400 and RFC 822), 33–68, 33–105
 EXPIRY alias file named parameter, 35–33
 expirysource channel option, 33–68, 33–105
 Fixup of
 passthrough/relay/submit channel options, 33–119
 Sieve setoperation does not affect, 5–62
 Folding of
 headerfold* channel options, 33–69
 headerlinelength channel option, 33–69
 LINELENGTH header trimming option, 33–157
 From:
 authrewrite channel option, 33–32, 33–63, 33–147
 AUTH_REWRITE mapping table, 33–33, 33–65, 33–148

-
- AUTH_REWRITE mapping table probe, 33–32, 33–64, 33–148
 - DSN human-readable part, 47–10
 - fullfromheader MSHTTP option, 29–8
 - MDNs, RETURN_PERSONAL option in disposition_option.opt, 47–21
 - message_hash_fields MTA option, 54–19
 - Fruit-of-the-day-warning:
 - fruits_size MTA option, 39–178
 - Fruit-of-the-day:
 - Example Sieve test on, 5–64
 - HTAB character
 - headerfoldpreserve channel option, 33–70
 - Importance:
 - alias_importance alias option, 35–17
 - Contrasted with Priority:, 42–5
 - IMPORTANCE alias file named parameter, 35–35
 - Mass mailings, 36–9
 - Relabelling from X-Priority:, 33–158
 - Label alignment, 33–69
 - Line folding
 - FOLDITEMS header trimming option, 33–157
 - Line wrapping
 - MIME parameter segmentation, 33–50, 33–54
 - List-*:
 - Adding via mgrpAddHeader group LDAP attribute, 39–139
 - Nested list definitions, 36–19
 - List-Archive:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - List-Help:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - List-id:
 - Disables sending back a vacation message, 5–45
 - List-Owner:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - List-Post:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - List-Subscribe:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - List-Unsubscribe:, 35–16, 35–34
 - Disables sending back a vacation message, 5–45
 - Logging in MTA transaction log
 - log_header MTA option, 39–261
 - transactionlog Sieve action, 39–232, 39–269
 - Maximum length of total in message
 - headercut channel option, 33–69
 - headerlimit channel option, 33–71
 - Message-Context:
 - Message Store message typing, header
 - Message Store msgasgtype option, 16–20
 - OUT-MESSAGE-CONTEXT conversion entry parameter, 38–11
 - Message-context:
 - IN-MESSAGE-CONTEXT conversion entry parameter, 38–9
 - Message-hash:, 33–90
 - Message-id:
 - identifiers switch of test -rewrite utility, 58–35
 - Altering via REVERSE mapping, 57–3
 - Default for duplicate Sieve test, 5–25
 - DSN human-readable part, 47–10
 - id_domain MTA option, 39–221
 - message_hash_fields MTA option, 54–19
 - Modification of, 39–63, 39–200
 - Modifying internal names in, 57–2
 - MTA options, 39–220
 - passyntaxerrors channel option, 33–68
 - message_hash_fields MTA option, 39–204
 - Mime-version:
 - Charset labelling forces addition of, 33–52
 - MR-Received:
 - max_mr_received_lines MTA option, 39–221
 - max_total_received_lines MTA option, 39–222
 - MT-Priority:
 - envelopetunnel channel option, 33–68
 - Original-recipient:
 - addrreturnpath channel option, 33–63
 - Defined in RFC 2298 (Message Disposition Notifications), 33–63
 - Personal names (RFC 822 phrases), 35–55
 - *personal* channel options, 33–41, 33–76
 - ldap_personal_name MTA option, 39–120
 - Postmaster, return_personal MTA option, 39–216
 - Position vis-à-vis message body
 - header* channel options, 33–69
 - Precedence:
 - alias_precedence alias option, 35–17
 - Contrasted with Priority:, 42–5
 - PRECEDENCE alias file named parameter, 35–35
 - use_precedence MTA option, 39–217
 - Preferred-language:
 - DISPOSITION_LANGUAGE mapping table probe, 47–17
 - language channel option, 33–72, 33–96

NOTIFICATION_LANGUAGE mapping table probe, 47–9
 Vacation message, Choice of body text, 39–127
 Vacation message, Choice of Subject:, 39–127
 Priority:
 *after channel options, 33–100
 *backoff channel options, 33–101
 alias_priority alias option, 35–17
 Defined in RFC 2156 (MIXER: Mapping between X.400 and RFC 822/MIME), 42–5
 Delay in delivery attempt, 33–100
 Effect on delivery retry frequency, 33–101
 Effect on MTA processing, 42–2, 42–5
 Effect on timing of generation of notification messages, 33–97
 Frequency of delivery reattempts, 33–101
 Job Controller delivery execution window, 42–16
 Job Controller handling, 42–5
 log_priority MTA option, 39–265
 Mass mailings, 36–9
 Overridden by MT-PRIORITY, 39–219
 Overridden by MT-PRIORITY, log_mtpriority MTA option, 39–264
 Overridden via setpriority Sieve action, 5–62
 PRIORITY alias file named parameter, 35–35
 profile SMS gateway_profile option, 53–7
 Relabelling from X-MSMail-Priority:, 33–158
 SMS messages, 53–8
 Processing of
 Channel options, 33–62
 fixsyntaxerrors channel option, 33–68
 inner channel option, 33–72
 limitheadertermination channel option, 33–73
 maxprocchars channel option, 33–90
 max_header_lines MTA option, 39–208
 passthrough/relay/submit channel options, 33–119
 relaxheadertermination channel option, 33–73
 Proxy-ip:, 57–3
 Received-SPF:, 33–145
 Received:
 "(original mail from sender-address)" comments, 33–74
 "by host-name" clause, Milter, 45–12
 "by host-name" clause, received_domain MTA option, 39–222
 "for recipient-address" clauses, 33–74
 "priority <mt-priority>" clause, 33–106, 33–130
 (envelope-sender (sender)) comments, 39–239
 (Forwarded-for: <source-ip>) clause, 57–3
 alias_noreceivedfor alias option, 35–19
 alias_noreceivedfrom alias option, 35–19
 alias_receivedfor alias option, 35–19
 alias_receivedfrom alias option, 35–19
 Axs:One PostedDate, 16–15
 CHECK_SOURCE TCP/IP-channel-specific option, 49–22
 conditionalpassthrough channel option, 33–120
 conditionalrelay channel option, 33–120
 Day of the week, 33–66
 deleteheader Sieve action cannot delete, 5–4
 Diagnosing .HELD files, 52–11
 DNS lookup, 33–138
 Format of, 33–74
 Format of date, 33–66
 Format of, Milter, 45–12
 from clause, XCLIENT SMTP command, 33–76, 33–132, 33–153
 generatereceivedheader MSHTTP option, 29–5
 host name comment, CHECK_SOURCE TCP/IP-channel-specific option, 49–22
 IDENT lookup, 33–138
 Mailing list postings, 35–38
 max_local_received_lines MTA option, 39–221
 max_received_lines MTA option, 39–221
 max_total_received_lines MTA option, 39–222
 Modifying internal names in, 57–2
 MTA options, 39–220
 MTA's own name appearing, max_local_received_lines MTA option, 39–221
 OpenDKIM, MAX_PREPEND_INDEX milter spamfilter option, 45–13
 received_version MTA option, 39–222
 Sieve filter access to, sieve_received MTA option, 39–225
 spamfilterN_received MTA options, 39–239
 state clause, convert/defragment, 52–3
 state clause, receivedstate channel option, 33–77
 Used for date in archiving, 45–9
 version Sieve filter environment item, 5–15
 Relabelling
 Header trimmming approach, 33–158
 MIME relabelling approach, 38–25
 MIME relabelling approach, RELABEL conversion entry parameter, 38–8
 Removing of
 deleteheader Sieve action, 5–25
 ldap_remove_header MTA option, 39–139
 Reply-to:

alias_nooriginator_reply alias option, 35–18
 alias_originator_reply alias option, 35–18
 ORIGINATOR_REPLY alias file named parameter, 35–37
 usereplyto channel option, 33–78
 Require-Recipient-Valid-Since:
 checkrrvs channel option, 33–35, 33–118
 Resent-*:
 Sieve filter redirect action, :resent
 and :noresent parameters, 5–41
 useresent channel option, 33–78
 Resent-Bcc:
 message_hash_fields MTA option, 54–19
 Resent-Cc:
 dropblank channel option, 33–68
 message_hash_fields MTA option, 54–19
 Resent-Date:
 sieve_redirect_add_resent MTA option, 39–225
 Resent-From:
 authrewrite channel option, 33–32, 33–63, 33–147
 AUTH_REWRITE mapping table, 33–33, 33–65, 33–148
 AUTH_REWRITE mapping table probe, 33–32, 33–64, 33–148
 message_hash_fields MTA option, 54–19
 sieve_redirect_add_resent MTA option, 39–225
 Resent-message-id:
 message_hash_fields MTA option, 54–19
 Resent-Sender:
 authrewrite channel option, 33–32, 33–63, 33–147
 AUTH_REWRITE mapping table, 33–33, 33–65, 33–148
 AUTH_REWRITE mapping table probe, 33–32, 33–64, 33–148
 Resent-To:
 dropblank channel option, 33–68
 message_hash_fields MTA option, 54–19
 sieve_redirect_add_resent MTA option, 39–225
 Return-path:
 addreturnpath channel option, 33–63
 Defined in RFC 2821 (SMTP), Section 4.4
 Trace Information, 33–63
 Passing to spam/virus filter package, 39–239
 Sender:
 Adding SMTP AUTH authenticated address, 44–14
 authrewrite channel option, 33–32, 33–63, 33–147
 AUTH_REWRITE mapping table, 33–33, 33–65, 33–148
 AUTH_REWRITE mapping table probe, 33–32, 33–64, 33–148
 Replace via FROM_ACCESS mapping table, 44–9
 Sensitivity:
 alias_sensitivity alias option, 35–17
 log_sensitivity MTA option, 39–267
 SENSITIVITY alias file named parameter, 35–35
 sensitivity* channel options, 33–107
 SMS messages, 53–9
 Size limits
 header_limit MTA option, 39–207
 max_header_blocks MTA option, 39–208
 max_header_lines MTA option, 39–208
 Solicitation:
 *nosolicit channel options, 33–124
 Spam-test:
 Example from SpamAssassin, 45–9
 Status:
 popstatuskludge POP option, 22–2
 Subject:
 addtag Sieve action, 5–18, 5–49
 alias_tag alias option, 35–23
 DSN human-readable part, 47–10
 DSNs generated by the MTA, 47–10
 IN-SUBJECT conversion entry parameter, 38–9
 ldap_add_tag MTA option, 39–139
 MDNs, 47–19
 MDNs, SUBJECT option in disposition_option.opt, 47–20
 message_hash_fields MTA option, 54–19
 SMS messages, text_to_subject gateway_profile option, 53–4
 SMS text charset, email_header_charset gateway_profile option, 53–5
 SMS text messages, charset, 53–7
 SMS text messages, parse_re_<n> gateway_profile options, 53–6
 TAG alias file named parameter, 35–40
 Tag on gatewayed SMS to email messages, in_re gateway_profile option, 53–5
 Tag on mailing list postings, 35–39
 Vacation message, ldap_autoreply_subject MTA option, 39–126
 Sun-Java-System-SMTP-Warning;, 33–133, 33–133
 Sun-ONE-SMTP-Warning;, 33–133
 Sun-One-SMTP-Warning;, 33–133
 To:

- "Recipients not specified: ;" or other empty group construct, 39–58
- alias_to alias option, 35–23
- dropblank channel option, 33–68
- DSN human-readable part, 47–10
- Generated when original message had no recipient header lines, 39–58
- Mailing list postings, TO named parameter, 35–40
- message_hash_fields MTA option, 54–19
- passyntaxerrors channel option, 33–68
- Use in Axs:One archiving, userheaderecipients Message Store archive option, 16–15
- To: Recipients not specified: ;
 - missingrecipientpolicy channel option, 33–39, 33–74
 - missing_recipient_group_text MTA option, 39–58
 - missing_recipient_policy MTA option, 39–59
- User-Agent:
 - MSIE, gzipattach MSHTTP option, 29–6
- Warnings-to:
 - use_warnings_to MTA option, 39–217, 39–217
- White space, 33–72
 - Terminating header, 33–73
- Wrapping of
 - LINELENGTH header trimming option, 33–157
- X-Accept-language:
 - DISPOSITION_LANGUAGE mapping table probe, 47–17
 - NOTIFICATION_LANGUAGE mapping table probe, 47–9
 - Vacation message, Choice of body text, 39–127
 - Vacation message, Choice of Subject:, 39–127
- X-Envelope-from:
 - MESSAGE-HEADER-FILE=2 conversion entry parameter, 38–10
- X-Envelope-to:, 33–75
 - MESSAGE-HEADER-FILE=2 conversion entry parameter, 38–10
- X-Forwarded-for:, 57–3
- X-Mailer:
 - xmailer MSHTTP option, 29–6
- X-MS-Exchange-Organization-Journal-Report:, 39–96, 39–204
 - In "capture :journal" 2007 format message copy, 54–16
- X-MS-Journal-Report:, 39–96, 39–204
 - In "capture :journal" 2003 format message copy, 54–14
 - In "capture :journal" 2007 format message copy, 54–16
- X-MSMail-Priority:
 - Relabelling to Priority:, 33–158
- X-Priority:
 - Relabelling to Importance:, 33–158
- X400-Received:
 - max_total_received_lines MTA option, 39–222
 - max_x400_received_lines MTA option, 39–222
- Header option file
 - test -rewrite utility, 58–35
- Header option files
 - Format, 33–156
 - Location, 33–156
- Header trim options, 33–155
 - ADD, 33–156
 - Defaults: field name, 33–156
 - FILL, 33–156
 - FOLDITEMS, 33–157
 - GROUP, 33–157
 - LINELENGTH, 33–157
 - MAXCHARS, 33–157
 - MAXIMUM, 33–157
 - MAXLINES, 33–157
 - Other: field name, 33–156
 - PRECEDENCE, 33–157
 - RELABEL, 33–158
- headerbottom channel option, 33–69
- headercut channel options, 33–69
- headerfoldpreserve channel option, 33–69
- headerfoldremove channel option, 33–69
- headerinc channel option, 33–69
- headerkeeporder channel option, 33–70
- headerlabelalignment channel option, 33–69
- headerlimit channel option, 33–71
- headerlineincrement channel option, 33–69
- headerlinelength channel option, 33–69
- headeromit channel option, 33–69
- headerread channel option, 33–70
 - Header option file, 33–155
 - Location, 33–156, 33–156
 - test -rewrite utility, 58–35
- headerset7 channel option, 33–53
- headerset8 channel option, 33–53
- headersetesc channel option, 33–53
- headertrailingpreserve channel option, 33–72
- headertrailingremove channel option, 33–72
- headertrim channel option, 33–70
 - Header option file, 33–155
 - Location, 33–156
 - Removing Received: header lines, 57–3
 - test -rewrite utility, 58–35
- header_733 channel option, 33–35

header_822 channel option, 33–35

header_limit MTA option, 39–207

- headerlimit channel option, 33–71

header_uucp channel option, 33–35

heartbeat Deployment Map option, 14–1

Held files

- alias_hold_* alias options, 35–16
- delivery_option clause, 39–93
- Hold channel, 52–10
- Hold channel and delivery_option clause, 39–93
- HOLD_LIST alias file named parameter, 35–35
- HOLD_MAPPING alias file named parameter, 35–35
- loopcheck channel option, 33–128
- mailDomainStatus of hold, 7–7, 39–145
- mailUserStatus of hold, 39–115
- max_local_received_lines MTA option, 39–221
- max_mr_received_lines MTA option, 39–221
- max_received_lines MTA option, 39–221
- max_total_received_lines MTA option, 39–222
- max_x400_received_lines MTA option, 39–222
- Releasing from hold channel, 52–11
- Sieve hold action, 5–51

held_sndopr MTA option, 39–220, 39–246

- Diagnosing .HELD files, 52–11
- max_mime_levels MTA option, 39–209
- max_mime_parts MTA option, 39–209

historical_time Dispatcher option, 41–4

history_file_directory SMS gateway option, 53–3

history_file_flush_period sms_gateway option, 53–3

history_file_flush_threshold sms_gateway option, 53–3

history_file_mode sms_gateway option, 53–3

history_file_rollover_period sms_gateway option, 53–4

history_to_return MTA option, 39–213

Hold channel, 52–10

- Configuration, 52–10
- Releasing messages from, 52–11

 - error_text_still_held MTA option, 39–162

- Routing to

 - delivery_options MTA option, 39–94, 39–94

- Runs reprocess channel program, 52–10

holdlimit channel option, 33–58, 33–89, 33–103, 33–113

- Diagnosing .HELD files, 52–11

hosteddomains MMP/imapproxy/popproxy/vdomain option, 27–15

hostname base option, 7–9

- Default for http.smtphost option, 29–10
- Direct LDAP alias lookups, 35–6
- ldap_local_host MTA option, 39–84, 39–98

host_hash_size MTA option, 39–179

htmlprocessor MSHTTP option, 29–13

- ICAP service use enabled, 32–1

httpproxyadmin MSHTTP option, 29–7

httpproxyadminpass MSHTTP option, 29–7, 29–7

I

ICAP, G–5

- HTML sanitization, 32–1
- See Spam/virus filter package integration, ICAP, 45–5
- Spam/virus filtering, 45–5

icapservice

- Options, 32–1

 - forcetelemetry, 32–1
 - server_host, 32–1
 - server_port, 32–1
 - service_name, 32–1

identnone channel option, 33–138

identnonelimited channel option, 33–138

identnonenumeric channel option, 33–138

identnonesymbolic channel option, 33–138

identtcp channel option, 33–138

identtcplimited channel option, 33–138

identtcpnumeric channel option, 33–138

identtcpsymbolic channel option, 33–138

idletimeout IMAP option, 21–11

idletimeout MSHTTP option, 29–12

idletimeout POP option, 22–3

id_domain MTA option, 39–221

- Limiting emission of internal host names, 57–2
- Local channel official_host_name, 52–2

ignoreencoding channel option, 33–46

ignoremessageencoding channel option, 33–46

ignoremultipartencoding channel option, 33–46

ignorerrvs channel option, 33–35, 33–118

image Dispatcher option, 41–4

IMAP

- ACL extension

 - Folder delivery, 33–42, 33–111

- ALERT

 - Overquota warning, 47–3
 - Password expiration warnings, 21–13
 - Password expiration warnings, pwchangeurl base option, 7–8
 - Password expiration, firstwarn pwexpirealert option, 21–13
 - Password expiration, metermaidtable pwexpirealert option, 21–13
 - Password expiration, viametermaid pwexpirealert option, 21–13
 - Quota warnings, 16–12

- quotanotification Message Store option, 16–12, 47–3
- serverdomainalert IMAP Proxy option, 27–22
- APPEND
 - Archiving, destination Message Store archive option, 16–14
- Autorestart
 - base.autorestart.enable option, 7–23
- CAPABILITY
 - capability IMAP proxy option, 27–9
 - Discovery of, 27–9
- Connection thread hold delay time
 - thresholddelay base option, 7–13
- Contexts
 - withinresolution IMAP option, 21–5
- Disconnect
 - Forcing via maxprotocolerrors IMAP option, 21–12
- DNS reverse lookup
 - dnsresolveclient base option, 7–16
- Extensions
 - ACL RIGHTS=tekx, capability_acl IMAP option, 21–5
 - ACL, capability_acl IMAP option, 21–5
 - ANNOTATE-EXPERIMENT-1, capability_annotate IMAP option, 21–5
 - AUTH=, Added to capability list by MMP, 27–9
 - BINARY, capability_binary IMAP option, 21–5
 - CATENATE, capability_catenate IMAP option, 21–5
 - CHILDREN, capability_children IMAP option, 21–5
 - CONDSTORE, capability_condstore IMAP option, 21–5
 - CONTEXT=SEARCH, capability_context_search IMAP option, 21–5
 - CONTEXT=SORT, capability_context_sort IMAP option, 21–5
 - CREATE-SPECIAL-USE, capability_create_special_use IMAP option, 21–8
 - ENABLE, capability_enable IMAP option, 21–6
 - ESEARCH, capability_earch IMAP option, 21–6
 - ESEARCH, capability_multisearch IMAP option, 21–7
 - ESORT, capability_esort IMAP option, 21–6
 - ID, capability_id IMAP option, 21–6
 - IDLE, capability_idle IMAP option, 21–6
 - LANGUAGE, capability_language IMAP option, 21–6
 - LANGUAGE, langlist MMP/IMAP proxy option, 27–16
 - LIST-STATUS, capability_list_status IMAP option, 21–6
 - LITERAL+, capability_literal IMAP option, 21–6
 - LOGIN-REFERRALS, capability_login_referrals IMAP option, 21–7
 - METADATA, capability_metadata IMAP option, 21–7
 - MULTISEARCH, capability_multisearch IMAP option, 21–7
 - NAMESPACE, capability_namespace IMAP option, 21–7
 - Non-standard, extra_capabilities IMAP option, 21–7
 - NOTIFY, capability_notify IMAP option, 21–7
 - QRESYNC, capability_qresync IMAP option, 21–7
 - QUOTA, capability_quota IMAP option, 21–8
 - SASL-IR, capability_sasl_ir IMAP option, 21–8
 - SEARCHRES, capability_searchres IMAP option, 21–8
 - SORT, capability_sort IMAP option, 21–8
 - SORT_DISPLAY, capability_sort_display IMAP option, 21–8
 - SPECIAL-USE, capability_special_use IMAP option, 21–8
 - STARTTLS, Added to capability list by MMP, 27–9
 - STARTTLS, capability_starttls IMAP option, 21–8
 - THREAD=ORDEREDSUBJECT, capability_thread_subject IMAP option, 21–9
 - THREAD=REFERENCES, capability_thread_references IMAP option, 21–9
 - UIDPLUS, capability_uidplus IMAP option, 21–9
 - UNSELECT, capability_unselect IMAP option, 21–9
 - URLAUTH, 49–7
 - URLAUTH, BURL_ACCESS mapping probes, 49–8
 - URLAUTH, capability_urlauth IMAP option, 21–9
 - WITHIN, capability_within IMAP option, 21–9
 - X-NETSCAPE, capability_x_netscape IMAP option, 21–9

X-ORCL-AS, capability_x_orcl_as IMAP option, 21-9
 X-SUN-IMAP, capability_x_sun_imap IMAP option, 21-9
 X-SUN-SORT, capability_x_sun_sort IMAP option, 21-10
 X-UNAUTHENTICATE, capability_x_unauthenticate IMAP option, 21-10
 XMSEARCH, capability_multisearch IMAP option, 21-7
 XREFRESH, capability_xrefresh IMAP option, 21-10
 XSENDER, capability_xsender IMAP option, 21-10
 XSERVERINFO, capability_xserverinfo IMAP option, 21-10
 XUM1, capability_xum1 IMAP option, 21-10

Flags
 immediateflagupdate IMAP option, 21-3
 Logging, log_conversion_tag MTA option, 39-255
 Logging, log_imap_flags MTA option, 39-262
 Set via Sieve filter at delivery time, 5-37
 shareflags Message Store
 privatesharedfolders option, 16-25
 Sieve filters, 5-1
 System flags begin with backslash, 5-37
 User, 5-38
 \Deleted, shareflags Message Store
 privatesharedfolders option, 16-25
 \Seen, Message expiration, 16-20
 \Seen, shareflags Message Store
 privatesharedfolders option, 16-25
 \Seen, store.expirerule.seen option, 16-20

IDLE
 immediateflagupdate IMAP option, 21-3

Last access time
 enablelastaccess base option, 7-9

Logging
 logauthsessionid option, 21-11
 logprotocolerrors IMAP option, 21-11

LSUB
 Unaffected by sharedfolders Message Store option, 16-8

msprobe probe of, 10-2

Options, 21-2, 21-3
 actionattributes, 21-11, 23-1
 actions, 21-11, 23-1
 adminbypassquota, 21-3
 allowanonymouslogin, 21-10
 banner, 21-10
 bgdecay, 7-4, 27-8
 bgexcluded, 7-4, 27-8
 bglinear, 7-4, 27-8
 bgmax, 7-4, 27-7
 bgpenalty, 7-4, 27-8
 capability_acl, 21-5
 capability_annotate, 21-5
 capability_binary, 21-5
 capability_catenate, 21-5
 capability_children, 21-5
 capability_condstore, 21-5
 capability_context_search, 21-5
 capability_context_sort, 21-5
 capability_create_special_use, 21-8
 capability_enable, 21-5
 capability_esearch, 21-6
 capability_esort, 21-6
 capability_id, 21-6
 capability_idle, 21-6
 capability_imap4, 21-6
 capability_imap4rev1, 21-6
 capability_language, 21-6
 capability_list_status, 21-6
 capability_literal, 21-6
 capability_login_referrals, 21-7
 capability_metadata, 21-7
 capability_multisearch, 21-7
 capability_namespace, 21-7
 capability_notify, 21-7
 capability_qresync, 21-7
 capability_quota, 21-8
 capability_sasl_ir, 21-8
 capability_searchres, 21-8
 capability_sort, 21-8
 capability_sort_display, 21-8
 capability_special_use, 21-8
 capability_starttls, 21-8
 capability_thread_references, 21-9
 capability_thread_subject, 21-9
 capability_uidplus, 21-9
 capability_unselect, 21-9
 capability_urlauth, 21-9
 capability_within, 21-9
 capability_xrefresh, 21-10
 capability_xsender, 21-10
 capability_xserverinfo, 21-10
 capability_xum1, 21-10
 capability_x_netscape, 21-9
 capability_x_orcl_as, 21-9
 capability_x_sun_imap, 21-9
 capability_x_sun_sort, 21-10
 capability_x_unauthenticate, 21-10
 connlimits, 27-10
 defaultdomain, 27-12

- domainallowed, 21–10, 27–13
- domainnotallowed, 21–11, 27–14
- enable, 21–3
- enableslport, 21–3
- enableuserlist, 21–11
- extra_capabilities, 21–7
- fixinternaldate, 21–3
- forcetelemetry, 21–11
- idletimeout, 21–11
- immediateflagupdate, 21–3
- legacy_proxyauth, 21–3
- logauthsessionid, 21–11
- logcommands, 21–4
- logprotocolerrors, 21–11
- logunauthsession, 21–13
- maxmessagesize, 21–4
- maxnoops, 21–4
- maxprotocolerrors, 21–11
- maxsearchmailboxes, 21–4
- maxsessions, 21–12
- maxthreads, 21–12
- numprocesses, 21–12
- plaintextmincipher, 21–12, 27–19
- polldelay, 21–4, 27–19
- See also obsoleteimap base option, 7–11
- slcachesize, 21–12
- sslnicknames, 21–12, 27–26
- sslport, 21–12
- sslusessl, 21–13
- submituser, 21–4, 39–68
- submituser, BURL usage, 49–11
- withinresolution, 21–5
- Password expiration alert
 - firstwarn pwexpirealert option, 21–13
 - metermaidtable pwexpirealert option, 21–13
 - viametermaid pwexpirealert option, 21–13
- Performance
 - preferpoll base option, 7–15, 27–20
- QUOTA extension
 - quotaroot Message Store messagetype
 - mtindex option, 16–22
- SEARCH
 - bodytextonly indexer option, 25–2
 - prefix_search indexer option, 25–3
 - substring_search indexer option, 25–2
 - suffix_search indexer option, 25–2
 - withinresolution IMAP option, 21–5
- Search queries
 - enable indexer option, 25–1
- SELECT
 - Unaffected by sharedfolders Message Store option, 16–8
- SSL

- enableslport IMAP option, 21–3
- sslport IMAP option, 21–12
- Startup, 21–3
- UID and UIDVALIDITY
 - log_mailbox_uid MTA option, 39–263
- IMAP commands
 - APPEND
 - adminbypassquota IMAP option, 21–3
 - maxmessagesize IMAP option, 21–4
 - Debugging
 - maparse keyword in debugkeys option, 27–12
 - search keyword in debugkeys option, 27–12
 - ESEARCH
 - maxsearchmailboxes IMAP option, 21–4
 - IDLE
 - capability_idle IMAP option, 21–6
 - LOGIN
 - broken_client_login_charset auth option, 12–2
 - NOOP
 - maxnoops IMAP option, 21–4
 - PROXYAUTH
 - legacy_proxyauth IMAP option, 21–3
 - STARTTLS
 - sslusessl option, 21–13
 - Xmailboxinfo MANAGEURL
 - folderurl base option, 7–8
 - Xserverinfo MANAGEACCOUNTURL
 - accounturl base option, 7–8
 - Xserverinfo MANAGEFILTERSURL
 - filterurl base option, 7–8
 - Xserverinfo MANAGELISTSURL
 - listurl base option, 7–8
- IMAP Proxy
 - Options, 27–3
 - authcachettl, 27–5
 - authenticationldapattributes, 27–6
 - authenticationserver, 27–6
 - backsideport, 27–6
 - banner, 27–7
 - canonicalvirtualdomaindelim, 27–9
 - capability, 27–9
 - connecttimeout, 27–10
 - connlimits, 27–10
 - crams, 27–11
 - debugkeys, 27–11
 - defaultdomain, 27–12
 - domainallowed, 27–13
 - domainnotallowed, 27–13
 - domainsearchformat, 27–14
 - hosteddomains, 27–15
 - langlist, 27–16
 - ldapcachesize, 27–16
 - ldapcachettl, 27–16

- ldappendingoplimit, 27–16
- ldaprefreshinterval, 27–17
- ldaptimeout, DEPRECATED, 27–17
- ldapurl, DEPRECATED, 27–17
- loglevel, 27–18, 39–72
- mailhostattrs, 27–18
- maxconcurrentconnectionattempts, 27–18
- preauth, 27–20
- preauthtimeout, 27–20
- replayformat, 27–20
- replaypass, 27–21
- requireauthenticationserver, 27–21
- restrictplainpasswords, 27–21
- searchformat, 27–21
- serverdownalert, 27–22
- ssladjustciphersuites, 7–11, 27–24
- sslbacksideport, 27–25
- sslcachedir, 7–3, 27–25
- storeadmin, 27–27
- storeadminpass, 27–27
- synldap, 27–27
- tcpaccess, 27–28
- tcpaccessattr, 27–28
- timeout, 27–28
- usergroupdn, DEPRECATED; see ugldapbasedn instead, 27–29
- ugldapbasedn instead, 27–29
- use_nslog, 27–29
- virtualdomaindelim, 27–29
- virtualdomainfile, DELETED; see vdomain options instead, 27–29
- imap: URLs
 - BURL_ACCESS mapping probes, 49–8
 - MTA URL types, 1–4
 - Only URL type to permit for BURL, 49–10
- imapadmin proxy option, 26–1
- imapadminpass proxy option, 26–1
- imapport proxy option, 26–1
- imaps: URLs
 - MTA URL types, 1–4
- imap_password MTA option, 39–68, 49–11
- imap_username MTA option, 39–68
 - BURL usage, 49–11
- imdbverify
 - crontab Scheduler task option, 8–4, 8–5
- imdbverify -s -m
 - crontab Scheduler task option, 8–4, 8–5
- imexpire job
 - crontab Scheduler task option, 8–3, 16–18
- immediateflagupdate IMAP option, 21–3
- implicitsaslexternal channel option, 33–151
- improute channel option, 33–38
 - improute_forward MTA option, 39–57
 - improute_forward MTA option
 - improute channel option, 33–39
- impurge daemon, 16–23
 - Disabling, 8–2
- imquotacheck utility
 - Notification that a Message Store user is overquota, 47–3
- ims-ms channel
 - Additional ims-ms_* channels
 - MESSAGE-SAVE-COPY mapping table, 54–5
- ims-ms channels, 51–1
 - Additional ims-ms_* channels, 51–4
 - alias_url0 MTA option, 51–3
 - backoff channel option, 51–1
 - Channel options
 - backoff, 51–1
 - defragment, 51–1
 - fileinto, 51–1
 - maxjobs, 51–1
 - notices, 51–1
 - official_host_name, 51–1
 - pool, 51–1
 - viaaliasrequired, 51–3
 - Channel program switches, 51–7
 - Configuration, 51–1
 - Additional ims-ms_* channels, 51–4
 - Debug, 51–7
 - Debug log files, 33–86
 - defragment channel option, 51–1
 - defragment option, 52–3
 - delivery_options MTA option, 51–3
 - Direct LDAP, 51–2
 - fileinto channel option, 51–1
 - loglevel option, 51–6
 - master_debug channel option, 51–6
 - maxjobs channel option, 51–1
 - Memory usage
 - dequeue_map MTA option, 39–172
 - Message Store stress, 42–4
 - notices channel option, 51–1
 - official_host_name channel option, 51–1
 - Options, 51–5
 - DEBUG, 51–6, 51–7
 - defragment, 52–3
 - DELIVER_THREADS, 51–6
 - FILEINTO, 51–6
 - FILEINTO, Interaction with fileinto channel option, 33–111
 - FILEINTO, Subaddresses in aliases, 35–45
 - LIFETIME_CAPACITY, 51–7
 - LOG_DEQUEUE_RATE, 51–7
 - ORIG_MAIL_ACCESS mapping table, 51–3
 - pool channel option, 51–1
 - syslogfacility option, 51–6

- viaaliasrequired channel option, 51–3
- ims5compat MSHTTP option, 29–5
- imsconnutil utility
 - enablelastaccess base option, 7–9
 - enableuserlist IMAP option, 21–11
 - enableuserlist MSHTTP option, 29–12
- imta_general_data MTA Tailor option (DELETED), 40–7
- imta_reverse_data MTA Tailor option (DELETED), 40–7
- imta_tmp MTA Tailor option
 - Effect on location of MTA database temp files, 40–4
- inactivity_time local_table MeterMaid option, 46–3
- includefinal channel option, 33–95
 - Recipient address reported in notifications, 47–6, 47–17
- include_connectioninfo MTA option, 39–189
- include_conversiontag MTA option, 39–190
 - *_ACCESS mapping table probes, 44–7
 - tag switch of test -rewrite utility, 58–38
 - CONVERSIONS mapping table, 38–2
 - Effect on FORWARD mapping table probe, 35–59
- include_mtpriority MTA option, 39–191
 - Effect on FORWARD mapping table probe, 35–59
- include_spares MTA option, 39–192
 - *_ACCESS mapping table probes, 44–7
- include_spares1 MTA option, 39–193
 - *_ACCESS mapping table probes, 44–7
- include_spares2 MTA option, 39–197
 - Effect on FORWARD mapping table probe, 35–59
- ldap_spare_N values, 39–125
- Indexer
 - Options, 25–1
 - bodytextonly, 25–2
 - enable, 25–1
 - port, 25–1
 - prefix_search, 25–3
 - server_host, 25–2
 - substring_search, 25–2
 - suffix_search, 25–2
 - timeout, 25–2
- indexer options
 - connecttimeout, 25–2
- indexeradmins Message Store option, 16–9
- Indexing and Search Service
 - Consulted by Messaging Server
 - indexer.enable option, 25–1
- indexsynclevel Message Store option, 16–6
- Initial configuration
 - dcroot base option, 7–16
 - defaultdomain base option, 7–4, 27–13
 - ens.enable, 61–1
 - hostname base option, 7–9
 - imap.enable, 21–3
 - mmp.enable, 27–5
 - mta.enable, 39–54
 - pop.enable, 22–1
 - purge.enable, 16–23
 - schedule.enable, 8–1
 - schedule.task:expire.crontab, 8–3, 16–18
 - schedule.task:msprobe.crontab, 8–3, 10–2
 - schedule.task:purge.crontab, 8–3, 16–24
 - schedule.task:snapshot.crontab, 8–4, 8–5
 - schedule.task:snapshotverify.crontab, 8–4, 8–5
 - store.enable, 16–4
 - store.serviceadmingroupdn, 16–13
 - ugldapbasedn base option, 7–14
 - ugldapbinddn base option, 7–14
 - ugldaphost base option, 7–14
 - ugldapport base option, 7–14
 - watcher.enable, 9–1
- inner channel option, 33–72
- innertrim channel option, 33–70
 - Header option file, 33–155
 - Location, 33–156
 - Removing Received: header lines, 57–3
 - test -rewrite utility, 58–35
- installedlanguages base option, 7–8
- instancename option, 2–1
- interfaceaddress channel option, 33–139
- interface_address Dispatcher option
 - See listenaddr Dispatcher option, 41–4
- interface_address Job Controller option
 - See listenaddr, 41–4
- interpretencoding channel option, 33–46
- interpretmessageencoding channel option, 33–46
- interpretmultipartencoding channel option, 33–46
- intext Message Store archive option, 16–15
- in_re gateway_profile option, 53–5
- IP address
 - Host's own
 - INTERNAL_IP mapping table, 44–6
 - In email address
 - Rewrite rule handling of, 34–8
 - Loopback
 - INTERNAL_IP mapping table, 44–6
- ipsecurity MSHTTP option, 29–8
- ipv6in base option, 7–17, 27–15
- ipv6in mmp option, 7–17, 27–15
- ipv6out base option, 7–18, 27–16
- ipv6out MMP option, 7–18, 27–16
- ipv6sortorder base option, 7–18

ipv6sortorder MMP option, 7–18
 ipv6usegethostbyname base option, 7–18
 ISO 8601 duration format, 1–3
 ISO 8601 format, 1–3
 ISO 8601 P format, 1–3
 ISO 8601 time format, 1–3
 ISS
 See Indexing and Search Service, 25–1
 ISS client
 See Indexer, 25–1
 it.com
 See Archive package integration, 45–9

J

JMQ

destinationtype notifytarget option, 24–4
 jmghost notifytarget option, 24–2
 jmqport notifytarget option, 24–2
 jmqpwd notifytarget option, 24–3
 jmqqueue notifytarget option, 24–3
 jmqtopic notifytarget option, 24–3
 jmquser notifytarget option, 24–3
 ldapdestination notifytarget option, 24–4
 msgtypes notifytarget option, 24–5
 persistent notifytarget option, 24–4
 priority notifytarget option, 24–4
 ttl notifytarget option, 24–4
 jmghost notifytarget option, 24–2
 jmqport notifytarget option, 24–2
 jmqpwd notifytarget option, 24–3
 jmqqueue notifytarget option, 24–3
 jmqtopic notifytarget option, 24–3
 jmquser notifytarget option, 24–3, 24–4
 Job Controller, 42–1
 *backoff channel options
 Scheduling of channel jobs, 33–101
 Autorestart
 autorestart.enable option, 7–23
 Changing options while running, 58–3
 channel_class, 42–17
 Checking that it is running, 42–18
 Configuration
 Default, 42–6
 Debugging
 debug option, 7–22, 42–10
 debug_flush MTA option, 39–73, 39–172
 Enabling, imsimta cache -change -global -
 debug=N, 58–4
 Example of enabling, 58–5
 job_pool, 42–17
 DEFAULT, Initial configuration, 42–6
 IMS_POOL, Initial configuration, 42–6
 SMTP_POOL, Initial configuration, 42–6

Log file
 cache -walk utility, 58–8
 debug Job Controller option, 7–22, 42–10
 msprobe probe of, 10–2
 Operation, 42–2
 Priority-based processing, 42–4
 Stress, 42–3
 Options, 42–9
 Changing values while running, 58–3
 channel_class, see Job Controller,
 channel_class, 42–17
 debug, 7–22, 42–10
 enable, 42–10
 interface_address, See listenaddr, 41–4
 job_limit, 42–11
 job_limit, maxjobs channel option, 33–105
 job_limit, Modified effect under stress, 42–4
 job_pool, See Job Controller, job_pool, 42–17
 listenaddr, 42–10
 master_command, 42–11
 max_cache_messages, 42–12
 max_cache_messages, cache -sync utility, 58–6
 max_cache_messages, Operation under
 stress, 42–3
 max_cache_messages, Overriding via imsimta
 cache -change -global -max_messages=N,
 58–4
 max_life_askwork, 42–13
 max_life_time, 42–13
 nonurgent_delivery, 42–15
 nonurgent_delivery, Example, 42–6
 normal_delivery, 42–16
 notice_time, Restricted: for future use, 42–13
 port, 42–13
 rebuild_in_order, Message replay example,
 54–5
 rebuild_parallel_channels, 42–13
 rebuild_parallel_channels, Override
 via imsimta cache -change -global -
 parallel_rebuild=N, 58–5
 Secret, 42–14
 slave_command, 42–14
 stressblackout, 42–4, 42–14
 stressfactor, 42–14
 stressfactor, Operation under stress, 42–4
 stressjobs, 42–15
 stressjobs, Operation under stress, 42–4
 stresstime, 42–14
 stresstime, Operation under stress, 42–4
 synch_time, 42–15
 synch_time, Hold channel, 52–11
 synch_time, Operation, 42–3

- synch_time, Retrieving messages from
- filter_discard, 52–8
- tcp_ports, 41–10, 42–15
- unstressfactor, 42–15
- unstressfactor, Operation under stress, 42–4
- unstressjobs, 42–15
- unstressjobs, Operation under stress, 42–4
- urgent_delivery, 42–16
- use_nslog, 27–29, 42–16
- Priority effects on delivery reattempt schedule, 33–101
- Processing pool
 - IMS_POOL dedicated to the ims-ms channel, 51–2
 - job_pool Job Controller group, 42–17
 - pool channel option, 33–106
 - Stopping from processing, 42–11
- Queue cache
 - max_cache_messages Job Controller option, 42–12
 - Operation under stress, 42–3
 - queue_cache_mode MTA option, 39–174
- Scheduling of channel jobs
 - *backoff channel options, 33–101
 - Solaris system parameters, 56–4
 - Startup, 39–54
- job_limit Job Controller option, 42–11
 - Modified effect under stress, 42–4
- job_pool group, 42–17
- journal_format MTA option, 39–96, 39–204
 - "capture :journal" message copies, 54–16

K

- keepmessagehash channel option, 33–90
- keylabel Message Store option, 16–12
- keypass Message Store option, 16–12
- kill utility
 - Effect on Dispatcher Worker Processes, 41–2

L

- L channel
 - Official host name
 - Default for received_domain MTA option, 39–222
 - official_host_name
 - ldap_local_host MTA option, 39–84, 39–98
- langdir MTA option, 39–154
 - Fallback location for return_*.txt files, 47–10
- langlist MMP/IMAP proxy option, 27–16
- Language
 - DSN generation, 47–12
 - Postmaster copy, 47–9
 - IMAP LANGUAGE extension

- langlist option, 27–16
- MDN generation, 47–20
- User preference
 - DISPOSITION_LANGUAGE mapping table, 47–17
 - ldap_preferred_language MTA option, 39–118
 - NOTIFICATION_LANGUAGE mapping table, 47–8
 - Postmaster copy of DSN, 47–9
 - preferredLanguage LDAP attribute, 39–118
 - Remote users, 47–12, 47–20
 - See also Header, Accept-language:, 47–8
 - See also Header, Preferred-language:, 47–8
 - See also Header, X-Accept-language:, 47–8
 - See also language channel option, 33–72, 33–96
 - Vacation message, Choice of body text, 39–127, 39–128
 - Vacation message, Choice of Subject:, 39–127, 39–127
- language channel option, 33–72, 33–96
- Language tag
 - CHARSET-CONVERSION mapping table, 38–19
 - Content-language: header line, 38–19
 - DISPOSITION_LANGUAGE mapping table, 47–17
- Encoded-words
 - CHARSET-CONVERSION mapping table, 38–19
- language channel option, 33–72, 33–96
- LDAP attributes with, 39–119
- ldap_preferred_language MTA option, 39–119
- NOTIFICATION_LANGUAGE mapping table, 47–8
- preferredLanguage LDAP attribute, 39–119
- sitelanguage base option, 7–9
- lastresort channel option, 33–62, 33–140
- LDAP ACI
 - PAB
 - Example, 39–182
- LDAP attributes
 - ACIs on, 39–113
 - Capture attribute, 54–6
 - Capture trigger attribute, 39–116
 - ldap_autoreply_addresses MTA option, 39–128, 39–128
 - ldap_filter_reference MTA option, 39–129
 - ldap_nosolicit MTA option, 39–119
 - ldap_parental_controls MTA option, 39–129
 - mailAutoReplyMode, 39–126
 - mailAutoReplySubject, 39–126
 - mailAutoReplyText, 39–127
 - mailAutoReplyTextInternal, 39–128

- mailDeliveryOption, 39–120
- mailForwardingAddress, 39–130
- mailProgramDeliveryInfo, 39–125
- mailSieveRuleSource, 39–129
- Message capture, 5–35
- PAB, 39–182
- preferredLanguage, 39–119
- Reassigning MTA interpretation of attributes, 39–103
- userPassword, 12–2
- vacationEndDate, 39–123
- vacationStartDate, 39–122
- Authentication library use, 39–103
- Caching of values, 39–152
- certSubjectDN, 7–24
- cn
 - Possible setting for ldap_personal_name MTA option, 39–120
- department
 - Example, 36–14
- Domain
 - aliasedObjectName, 7–6, 39–143
 - associatedDomain, 39–82, 39–142
 - Disk quota, 39–147
 - domainUidSeparator, Default for ldap_domain_attr_uid_separator MTA option, 7–6, 39–144
 - inetCanonicalDomainName, Default for ldap_domain_attr_canonical MTA option, 39–143
 - inetDomainBaseDn, Default for ldap_domain_attr_basedn MTA option, 7–6, 39–143
 - inetDomainMailserv, 39–143
 - inetDomainSearchFilter, canonicalsearchfilter auth option, 12–1
 - inetDomainSearchFilter, Possible value for ldap_attr_domain_search_filter MTA option, 39–82, 39–88, 39–142
 - inetDomainSearchFilter, searchfilter auth option, 12–1
 - inetDomainStatus, 7–7, 39–145
 - ldap_domain_attr_capture MTA option, 39–148
 - ldap_domain_attr_creation_date MTA option, 39–151
 - ldap_domain_attr_default_mailhost MTA option, 39–124, 39–147
 - ldap_domain_attr_disk_quota, Domain LDAP attribute to override defaultmailboxquota, 16–11
 - ldap_domain_attr_message_quota, Domain LDAP attribute to override defaultmessagequota, 16–11
 - ldap_domain_attr_optinN MTA option, 39–146
 - ldap_domain_attr_sourceblocklimit MTA option, 39–149
 - ldap_domain_attr_source_channel MTA option, 39–150
 - ldap_domain_attr_subaddress MTA option, 39–144
 - mailAccessProxyPreAuth, 27–20
 - mailAllowedServiceAccess, TCP wrapper access filters, 6–2
 - mailDomainAllowedServiceAccess, Authentication library use, 39–103
 - mailDomainAllowedServiceAccess, TCP wrapper access filter, 6–7
 - mailDomainAllowedServiceAccess, TCP wrapper access filters, 6–8
 - mailDomainCatchallAddress default for ldap_domain_attr_catchall_address, 39–149
 - mailDomainCatchallMapping default for ldap_domain_attr_catchall_mapping, 39–149
 - mailDomainConversionTag, 39–146, 39–146
 - mailDomainMsgMaxBlocks, 39–145
 - mailDomainReportAddress default for ldap_domain_attr_report_address, 39–148
 - mailDomainSieveRuleSource default for ldap_domain_attr_filter, 39–148
 - mailDomainSieveRuleSource, Sieve hierarchy, 5–65
 - mailDomainStatus, 7–7, 39–145
 - mailDomainStatus, imquotacheck utility setting to overquota, 7–7, 39–145
 - mailRoutingHosts, Default for ldap_domain_attr_routing_hosts MTA option, 39–144
 - mailRoutingSmartHost, Default for ldap_domain_attr_smarthost MTA option, 39–144
 - Message quota, 39–148
 - objectClass, 39–114
 - preferredMailHost, Possible value for ldap_domain_attr_default_mailhost MTA option, 39–147
 - Spam/virus opt-in, 39–146
 - sunPreferredDomain, 39–81, 39–142
 - Example of list in external LDAP directory, 36–14
 - expandable
 - Default for ldap_expandable MTA option, 39–140

Group

- delivery_options, Default for
 - ldap_delivery_option MTA option, 39–120
 - expandable, expn* channel options, 33–126
 - ldap_group_status MTA option, 39–115
 - ldap_personal_name MTA option, 39–120
 - mailAutoReplyMode, Default for
 - ldap_autoreply_mode, 39–126
 - mailAutoReplySubject, Default for
 - ldap_autoreply_subject MTA option, 39–126
 - mailAutoReplyTimeout, Default for
 - ldap_autoreply_timeout MTA option, 39–128
 - mailDeferProcessing, Default for
 - ldap_reprocess MTA option, 39–130
 - mailForwardingAddress, Default for
 - ldap_forwarding_address MTA option, 39–130
 - mailHost, Default for ldap_mailhost MTA option, 39–124
 - mailSieveRuleSource, ACI, 39–129
 - mailSieveRuleSource, Default for ldap_filter MTA option, 39–129
 - mailSieveRuleSource, Sieve hierarchy, 5–65
 - memberURL, Default for ldap_group_url2, 39–135
 - mgmanMemberVisibility, expn* channel options, 33–126
 - mgrpAddHeader, Default for
 - ldap_add_header MTA option, 39–139
 - mgrpAuthPassword, Default for
 - ldap_auth_password, 39–133
 - mgrpDeliverTo, Default for ldap_group_url1, 39–135
 - mgrpJettisonBroadcasters, Default for
 - ldap_jettison_url MTA option, 39–131
 - mgrpJettisonDomain, Default for
 - ldap_jettison_domain MTA option, 39–131
 - mgrpLastAccessTime, Default for
 - ldap_group_ldap_access_time, 39–134
 - mgrpListTag, Default for ldap_add_tag MTA option, 39–139
 - mgrpListTag, Language-tag, 39–139
 - mgrpModerator, Default for
 - ldap_moderator_url, 39–134
 - mgrpRemoveHeader, Default for
 - ldap_remove_header MTA option, 39–139
 - mgrpUniqueID, Additional
 - mgrpBroadcasterPolicy values, 39–132
 - objectClass, 39–114
 - preferredLanguage, ldap_preferred_language MTA option, 39–118
 - uniqueMember, Default for ldap_group_dn, 39–135
 - vacationEndDate, Default for ldap_end_date MTA option, 39–123
 - vacationStartDate, Default for ldap_start_date MTA option, 39–122
- ## Head-of-household controls
- ldap_filter_reference MTA option, 39–129
 - ldap_filter_reference MTA option, Sieve hierarchy, 5–65
 - ldap_hoh_filter MTA option, Sieve hierarchy, 5–65
 - ldap_parental_controls MTA option, 39–129
- ## inetDomainStatus
- Default for ldap_domain_attr_status MTA option, 7–7, 39–145
- ## inetMailGroupStatus
- Default for ldap_group_mail_status MTA option, 39–115
 - Deleted or removed,
 - error_text_deleted_group MTA option, 39–162
 - Disabled, error_text_disabled_group MTA option, 39–162
 - Inactive, error_text_inactive_group MTA option, 39–162
 - Supported values, 39–115
 - Values that disable vacation message generation, 5–45
- ## inetMailUser
- searchfilter default, 39–90
- ## inetUserStatus
- Authentication library use, 39–103
 - Default for ldap_user_status MTA option, 39–114
 - Deleted or removed, error_text_deleted_user MTA option, 39–162
 - Inactive, error_text_inactive_user MTA option, 39–162
 - Overquota, error_text_over_quota MTA option, 39–161
- ## Language-tag
- language channel option, 33–72, 33–96
 - ldap_autoreply_addresses MTA option, 39–128, 39–128
 - mailAutoReplySubject, 39–127
 - mailAutoReplyText, 39–127
 - mailAutoReplyTextInternal, 39–128
 - mgrpListTag, 39–139
 - preferredLanguage, 39–119
- ## ldapdestination notifytarget option, 24–4
- ## listID
- Example, 36–12
- ## mail
- Address reversal, reverse_url filter, 35–49

FROM_ACCESS mapping table use, 44–14
 Head-of-household use, 5–73
 Mailing list membership definitions, 36–10
 Omitting from group definition, 36–18
 Presence on group LDAP entry, 36–16
 SASL library use, 44–14
 Typically requested in LDAP URL alias lookups, 35–41
 uniqueMember group members, 36–10

mailAccessProxyReplay, 27–21

mailAllowedServiceAccess
 Authentication library use, 39–103
 SMTP AUTH effect, 49–52

mailAlternateAddress
 Address reversal, 35–49

mailAntiUBEService, 39–121

mailAutoReplyMode
 vacation message format, 47–8

mailAutoReplySubject
 Language-tag, 39–127

mailAutoReplyText
 Language-tag, 39–127
 Vacation message not generated, 5–46

mailAutoReplyTextInternal
 Default for ldap_autoreply_text_internal MTA option, 39–128
 Language-tag, 39–128
 Vacation message not generated, 5–46

mailAutoReplyTimeout
 Vacation message not generated, 5–45

mailAutoReplyTimeOut
 vacation_maximum_timeout MTA option, 39–66, 39–102
 vacation_minimum_timeout MTA option, 39–67, 39–102

mailConversionTag, 39–123

mailDeferProcessing, 39–184
 Example on large mailing list, 36–20
 Mass mailings, 36–21

mailDeliveryOption, 39–13, 39–184
 Custom values for custom ims-ms_* channel delivery, 51–5
 delivery_options interpretation, 39–92
 mailbox delivery via ims-ms channel, 51–4

mailDeliveryOptions
 forward, sieve_user_carryover MTA option, 39–101, 39–225

mailDomainMsgMaxBlocks
 Address reversal, 35–50
 error_text_list_block_over MTA option, 39–159
 error_text_user_block_over MTA option, 39–160

mailDomainReportAddress
 Address reversal, 35–50

mailDomainStatus
 Default for ldap_domain_attr_mail_status MTA option, 7–7, 39–145
 Hold channel, 52–10
 Hold channel, Releasing messages, 52–11
 Overquota, error_text_over_quota MTA option, 39–161
 spooftempfail POP Proxy option, 27–23
 Values that disable vacation message generation, 5–45

mailEquivalentAddress
 Address reversal, 35–49

mailEventNotificationDestination, 24–4

mailHost
 aliasdetourhost override of, 33–30, 33–60
 dequeueremoveroute channel option, 33–37
 enqueueeremoveroute channel option, 33–38
 Error text when a user entry that needs a mailHost, lacks one, 39–158
 IMAP AUTHURL use, 49–9
 proxytrustmailhost base option, 7–16
 storehostlist Proxy option, 26–2

Mailing lists
 mail, Fetched during head of household Sieve filter lookups, 39–130
 mailHost,
 ldap_domain_attr_default_mailhost MTA option, 39–147
 mailSieveRuleSource, Fetched during head of household Sieve filter lookups, 39–130
 mgrpDigestInterval, 39–138
 uniqueMember, 36–11

mailMsgMaxBlocks
 Address reversal, 35–49
 Notification messages, 47–25

mailMsgQuota
 Text of quotaexceededmsg message, 16–13, 17–1

mailQuota
 Text of quotaexceededmsg message, 16–13, 17–1

mailRoutingAddress, 39–119

mailRoutingHosts
 route_to_routing_host MTA option, 39–101

mailRoutingSmartHost
 enqueueeremoveroute channel option, 33–38

mailSieveRuleSource
 Head-of-household use, 5–73

mailSMTPSubmitChannel, 33–82, 33–128, 33–155, 39–103
 Use with FUTURERELEASE, 49–12

mailUserStatus
 Authentication library use, 39–103
 Default for `ldap_user_mail_status` MTA option, 39–114
 Deleted or removed, `error_text_deleted_user` MTA option, 39–162
 Hold channel, 52–10
 Hold channel, Releasing messages, 52–11
 Inactive, `error_text_inactive_user` MTA option, 39–162
 Overquota, `error_text_over_quota` MTA option, 39–161
 Set to overquota by Message Store, 16–7
 SMTP AUTH effect, 49–52
`spooftempfail` POP Proxy option, 27–23
 Values that disable vacation message generation, 5–45

memberOf
 Example, 36–12

memberURL
 Example, 36–9
 Example in meta-list, 36–14
 Mailing list membership, 36–18
 Mass mailings, 36–8
`process_substitutions` MTA option, 39–100

Message size limits
`ldap_sourceblocklimit` MTA option, 39–118

mgmanhidden, 29–4

mgmanMemberVisibility
 Default for `ldap_expandable` MTA option, 39–140

mgrpAddHeader
 Default for `ldap_add_header` MTA option, 39–139

mgrpAllowedBroadcaster, 39–184
 Default for `ldap_auth_url` MTA option, 39–132
 Example, 36–20
 Multiple values ORed, 36–2
`process_substitutions` MTA option, 39–100

mgrpAllowedDomain
 Default for `ldap_auth_domain` MTA option, 39–133
 Multiple values ORed, 36–2

mgrpAuthPassword
 Example, 36–20

mgrpBroadcasterPolicy, 39–132
 Example, 36–20

mgrpDelayNotifications, 39–138

mgrpDeliverTo
 Mass mailings, 36–8
`process_substitutions` MTA option, 39–100

mgrpDisallowedBroadcaster
 Default for `ldap_cant_url` MTA option, 39–132
`process_substitutions` MTA option, 39–100

mgrpDisallowedDomain
 Default for `ldap_cant_domain` MTA option, 39–132

mgrpErrorsTo
 Analogous to `alias_envelope_from` alias option, 35–14
 Mailing list vs. group, 36–16, 36–16
 Setting to / value, 36–17

mgrpMaxMessagesPerDay
 Default for
`ldap_maximum_messages_per_day` MTA option, 39–133

mgrpModerator
`process_substitutions` MTA option, 39–100

mgrpMsgMaxSize
 Default for `ldap_maximum_message_size` MTA option, 39–133
`error_text_list_block_over` MTA option, 39–159
`error_text_user_block_over` MTA option, 39–160

mgrpMsgPrefixText
 Default for `ldap_prefix_text` MTA option, 39–140

mgrpMsgRejectAction, 39–131

mgrpMsgRejectText, 39–131

mgrpMsgSuffixText
 Default for `ldap_suffix_text` MTA option, 39–140

mgrpRejectText, 39–131

mgrpRemoveHeader
 Default for `ldap_remove_header` MTA option, 39–139

mgrpUniqueId
 Default for `ldap_list_id` MTA option, 39–131

mgrpURLResultMapping
 Example, 36–13

msgVanityDomain, 35–8
`domain_match_url` MTA option, 39–80
 MTA use of, 39–103
 Multi-purpose use, 39–102

objectClass
`ldap_objectclass` MTA option, 39–114

PAB
`displayName`, 5–31
`memberOfPiGroup`, 5–31
`piEmail*`, 5–31
`piEntryID`, 5–31

Parental controls
`ldap_filter_reference` MTA option, 39–129

- ldap_filter_reference MTA option, Sieve hierarchy, 5–65
- ldap_hoh_filter MTA option, Sieve hierarchy, 5–65
- ldap_parental_controls MTA option, 39–129
- preferredLanguage
 - Address reversal, 35–49
 - DISPOSITION_LANGUAGE mapping table probes, 39–119
 - Effect on mgrpListTag, 39–139
 - NOTIFICATION_LANGUAGE mapping table probes, 39–119
- Sieve filters
 - ldap_filter_reference MTA option, 39–129
 - mailSieveRuleSource, 39–129
- SMSdomain, 36–13
- smsID, 36–13
- Source channel switch
 - ldap_source_channel MTA option, 39–118
- uid
 - Authentication library use, 39–103
 - Canonical authenticated identity in BURL_ACCESS probes, 49–8
 - Illegal characters, Error text, 39–158
 - Invalid characters in, 39–99
 - ldap_domain_attr_uid_separator MTA option, 7–6, 39–144
 - ldap_uid MTA option, 39–115
 - Length limit, 39–99, 39–116
 - Logging via log_uid MTA option, 39–269
 - mailbox delivery via ims-ms channel, 51–4
 - Must be single-valued, 39–116
- uniqueMember, 36–10
 - group_dn_template MTA option, 39–14
 - Interpretation affected by group_dn_template MTA option, 39–96
- User
 - Capture attribute, ACI, 54–6
 - cn, Used by MSHTTP if fullfromheader MSHTTP option set, 29–8
 - delivery_options, Default for ldap_delivery_option MTA option, 39–120
 - extrauserldapattrs MSHTTP option, 29–8
 - inetUserStatus, Authentication library use, 39–103
 - ldapdestination, 24–2
 - ldap_autoreply_addresses MTA option, ACI on, 39–128
 - ldap_filter_reference MTA option, Sieve hierarchy, 5–65
 - ldap_personal_name MTA option, 39–120
 - ldap_preferred_country MTA option, 39–119
 - mail, Default for ldap_auth_attr_sender MTA option, 39–151
 - mail, Default for ldap_default_attr MTA option, 39–86
 - mail, Fetched during head of household Sieve filter lookups, 39–130
 - mail, Head-of-household purposes, 39–97, 39–141
 - mailAllowedServiceAccess, 27–28
 - mailAllowedServiceAccess, altservice MSHTTP option, 29–11
 - mailAllowedServiceAccess, Authentication library use, 39–103
 - mailAllowedServiceAccess, TCP wrapper access filter, 6–7
 - mailAllowedServiceAccess, TCP wrapper access filters, 6–2, 6–8
 - mailAlternateAddress, Matching for vacation message generation, 5–44
 - mailAutoReply*, Vacation message generation, 5–44
 - mailAutoReplyMode, ACI, 39–126
 - mailAutoReplyMode, Default for ldap_autoreply_mode, 39–126
 - mailAutoReplySubject, ACI, 39–126
 - mailAutoReplySubject, Default for ldap_autoreply_subject MTA option, 39–126
 - mailAutoReplyText, ACI, 39–127
 - mailAutoReplyText, Default for ldap_autoreply_text MTA option, 39–127
 - mailAutoReplyTextInternal, ACI, 39–128
 - mailAutoReplyTextInternal, vnd.sun.autoreply-internal Sieve environment item, 5–15
 - mailAutoReplyTimeout, ACI, 39–129
 - mailAutoReplyTimeout, Default for ldap_autoreply_timeout MTA option, 39–128
 - mailCaptureInternet (site-defined), 5–35
 - mailConversionTag, MESSAGE-SAVE-COPY mapping table detection of, 54–5
 - mailDeferProcessing, Default for ldap_reprocess MTA option, 39–130
 - mailDeliveryOption value of program, Pipe options, 52–15
 - mailDeliveryOption, nomail, 36–22
 - mailDeliveryOption, Pipe channel, 52–14
 - mailDeliveryOptions, Forwarding user's mail, 35–59
 - mailEquivalentAddress, Matching for vacation message generation, 5–44
 - mailForwardingAddress, ACI, 39–130

mailForwardingAddress, Default for ldap_forwarding_address MTA option, 39–130
 mailForwardingAddress, Forwarding user's mail, 35–59
 mailHost, aliasdetourhost override of, 33–30, 33–60
 mailHost, checkmailhost Message Store option, 16–5
 mailHost, Default for ldap_auth_attr_mail_host MTA option, 39–151
 mailHost, Default for ldap_mailhost MTA option, 39–124
 mailHost, Fallback if MSHTTP option smtp host not responding, 29–10
 mailHost, ldap_domain_attr_default_mailhost MTA option, 39–147
 mailHost, ldap_host_alias_list base option, 7–9
 mailHost, mailhostattrs mmp/imaproxy/popproxy/vdomain option, 27–18
 mailMsgQuota, Message type example, 16–21
 mailMsgQuota, Override defaultmessagequota store option, 16–11
 mailMsgQuota, Per message type, 16–22
 mailProgramDeliveryInfo, \$P substitution in LDAP URLs, 1–7
 mailProgramDeliveryInfo, Pipe channel, 52–14
 mailQuota, Message type example, 16–21
 mailQuota, Override defaultmailboxquota store option, 16–11
 mailQuota, Per message type, 16–22
 mailSieveRuleSource, 33–109
 mailSieveRuleSource, ACI, 39–129
 mailSieveRuleSource, Default for ldap_filter MTA option, 39–129
 mailSieveRuleSource, Fetched during head of household Sieve filter lookups, 39–130
 mailSieveRuleSource, Head-of-household purposes, 39–97, 39–141
 mailSieveRuleSource, Sieve hierarchy, 5–65
 mailSMTPSubmitChannel, 33–82, 39–103
 mailSMTPSubmitChannel, Default for ldap_auth_attr_submit_channel MTA option, 39–151
 mailSMTPSubmitChannel, saslswitchchannel channel option, 33–20
 mailSMTPSubmitChannel, SMTP long lines, 33–134
 mailUserStatus, Authentication library use, 39–103
 mailUserStatus, Set to overquota by Message Store, 16–7
 msgVanityDomain, 34–30, 35–8, G–11
 objectClass, 39–114, 39–114
 Passed to third-party authentication server, authenticationldapattributes auth option, 27–6
 preferredLanguage, ldap_preferred_language MTA option, 39–118
 preferredLanguage, ldap_spare_4, ldap_spare_5, ldap_spare_6 values, 39–125
 psroot, 5–30
 uid, \$M substitution in LDAP URLs, 1–7
 uid, Authentication library use, 39–103
 uid, canonicalsearchfilter auth option, 12–1
 userCertificate, verifycert Base certmap option, 7–23
 userPassword, 39–103
 userPassword, ACI on, 12–2
 userPassword, authcachettl timeout on caching, 27–5
 userPassword, Correcting noncompliant password entry, 12–2
 userPassword, crams mmp/imaproxy/popproxy/vdomain option, 27–11
 vacation*, Vacation message generation, 5–44
 vacationEndDate, Default for ldap_end_date MTA option, 39–123
 vacationStartDate, Default for ldap_start_date MTA option, 39–122
 User-modifiable, 39–113
 ldap_autoreply_addresses MTA option, 39–128
 mailAutoReplyMode, 39–126
 mailAutoReplySubject, 39–126
 mailAutoReplyText, 39–127
 mailAutoReplyTextInternal, 39–128
 mailAutoReplyTimeout, 39–129
 mailDeliveryOption, 39–120
 mailForwardingAddress, 39–130
 mailProgramDeliveryInfo, 39–125
 mailSieveRuleSource, 39–129
 preferredLanguage, 39–119
 userPassword, 39–103
 vacationEndDate, 39–123
 vacationStartDate, 39–122
 userID
 Synonym for uid, 39–99
 userPassword
 Authentication library use, 39–103
 Vacation message generation

- Address recognition,
 - ldap_autoreply_addresses MTA option, 39–128
 - mailAutoReplyMode, 39–126
 - mailAutoReplySubject, 39–126
 - mailAutoReplyText, 39–127
 - mailAutoReplyTextInternal, 39–128
 - mailAutoReplyTimeout, 39–128
 - mailSieveRuleSource, 39–129
 - preferredLanguage, 39–127, 39–128
 - preferredLanguage, mailAutoReplySubject language-tag, 39–126
- vacationEndDate
 - Current date comparison for vacation message generation, 5–45
- vacationStartDate
 - Current date comparison for vacation message generation, 5–45
- LDAP bind and connect MTA options, 39–76
- LDAP external directory lookups
 - Example, 36–15, 49–41
 - extldap: and extldaps: URLs, 1–4
 - ldap_ext_host MTA option, 39–182
 - ldap_ext_max_connections MTA option, 39–182
 - ldap_ext_password MTA option, 39–182
 - ldap_ext_port MTA option, 39–182
 - ldap_ext_username MTA option, 39–182
 - MTA options, 39–181
- LDAP lookups
 - Debugging
 - mm_debug MTA option, 39–74
 - ldap: and ldaps: URLs, 1–4
 - Mapping tables, 37–13
 - MMP POP and IMAP proxy
 - Performance, 27–27
 - Performance impact, 56–2
 - Rewrite rules, 34–22
- LDAP object classes
 - certificationauthority, 30–2, 30–2
 - Domain
 - inetDomain, 7–8, 39–83, 39–89
 - inetdomainalias, 7–8, 39–83, 39–89
 - sunManagedOrganization, 7–8, 39–83, 39–89
 - Group
 - inetLocalMailRecipient, iMS 5.0 schema, 39–89
 - inetmailgroup, iMS 5.0 schema, 39–89
 - inetmailgroup, SIMS 4.0 schema, 39–89
 - inetMailRouting, SIMS 4.0 schema, 39–89
 - ldap_group_object_classes MTA option, 39–89
 - mailGroup, NMS 4.1 schema, 39–89
 - inetOrgPerson
 - Defined in RFC 2798, 39–92, 39–102
 - ldap_objectclass MTA option, 39–114
 - msgCRLMappingTable, 30–5
 - PiTypeGroup, 5–31
 - User
 - Authentication purposes, searchfilter auth option, 39–91
 - Client certificate, filtercomps certmap option, 7–23
 - inetLocalMailRecipient, iMS 5.0 schema, 39–90
 - inetMailRouting, SIMS 4.0 schema, 39–90
 - inetMailUser, 36–10
 - inetmailuser, canonicalsearchfilter Auth option, 12–1
 - inetmailuser, Default searchfilter for authentication, 12–1
 - inetmailuser, iMS 5.0 schema, 39–90
 - inetmailuser, searchfilter auth option, 39–90
 - inetmailuser, SIMS 4.0 schema, 39–90
 - inetOrgPerson, 36–10
 - inetOrgPerson, Defined in RFC 2798, 39–92, 39–102
 - ldap_user_object_classes MTA option, 39–90
 - mailRecipient, NMS 4.1 schema, 39–90
 - nsMessagingServerUser, NMS 4.1 schema, 39–90
- LDAP PAB MTA options, 39–182
- LDAP schema, 27–15, 39–102
 - ACIs
 - mailAutoReplyTimeout attribute, 39–129
 - User-modifiable LDAP attributes, 39–113
 - Base DN for domain portion of the DIT
 - ldap_domain_root MTA option, 39–83, 39–89
 - Base DN for the user portion of the DIT
 - ldap_user_root MTA option, 39–87, 39–91
 - Default for group objectClasses, 39–89
 - Default for user objectClasses, 39–90
 - Extending, 7–24
 - iMS 5.0
 - ldap_schematag MTA option, 39–90
 - ldap_schemalevel base option, 7–5, 39–90
 - ldap_schematag MTA option, 39–90
 - MTA options, 39–88
 - NMS 4.1
 - ldap_schematag MTA option, 39–90
 - Renaming attributes of, 39–151
 - RFC 2798, 39–102
 - SIMS 4.0
 - ldap_schematag MTA option, 39–90
- LDAP server
 - Connection

- ldaprefreshinterval mmp/imaproxy/
popproxy option, 27-17
- Performance
 - ldap_domain_known_attributes MTA option,
7-6, 39-83
- Problems
 - Authentication server unavailable, 49-53
- Timeout on modifications
 - ldapmodifytimeout base option, 7-10
- Timeout on queries
 - ldapsearchtimeout base option, 7-10
 - ldaptimeout mmp/imaproxy/popproxy
option (DEPRECATED), 27-17
- LDAP StartTLS
 - ldaprequiretls base option, 7-15
- LDAP URL
 - Length limit, 37-15
 - See Length limits in configuration, 35-7
 - Mapping table substitution, 37-13
 - max_urls MTA option, 39-78
 - Quoting (encoding) requirements, 37-14
 - Recursive references
 - max_urls MTA option, 39-78
 - Rewrite rule substitution, 34-22
 - Substitution sequences, 1-5
 - \$A, Used in group_dn_template value, 36-11
 - \$A, Used in GROUP_TEMPLATES mapping
table template, 36-12
 - \$B, Used in GROUP_TEMPLATES mapping
table template, 36-12
 - \$S, Example in meta-list, 36-14
 - process_substitutions MTA option, 39-99
 - Special interpretation in
spamfilterN_action_M MTA options, 39-236
- Syntax
 - Alias value (in alias file or alias database),
35-41
 - alias_urlN MTA options, 35-6
 - ldap_default_attr MTA option, 39-86
- ldap: URLs
 - ldap:///V?\$N?sub?\$R
 - reverse_url option's default value, 39-88
 - ldap:///V?*?sub?\$R
 - alias_url0 option's default value, 39-86
 - MTA URL types, 1-4
 - Syntax, 34-22
 - Substitution sequences, 1-5
- ldapaddresssearchattrs MSHTTP option, 29-5
- ldapbasedn PAB option, 59-1
- ldapbinddn PAB option, 59-2
- ldapcachesize MMP/IMAP Proxy/POP Proxy/
SUBMIT Proxy/vdomain option, 27-16
- ldapcachettl MMP et al. option, 27-16
- ldapcheckcert base option, 7-9
- ldapconnecttimeout base option, 7-10
 - Direct LDAP alias lookups, 35-5
- ldapdestination notifytarget option, 24-4
- ldaphost PAB option, 59-2
 - ldap_pab_host MTA option override for MTA
PAB query purposes, 39-183
- ldapmodifytimeout base option, 7-10
- ldappasswd PAB option, 59-2
- ldappendingoplimit IMAP Proxy, POP Proxy, and
MMP option, 27-16
- ldappoolrefreshinterval base option, 7-10
- ldapport PAB option, 59-2
- ldaprefreshinterval mmp/imaproxy/popproxy
option, 27-17
- ldaprequiretls base option, 7-15
 - Direct LDAP alias lookups, 35-5
 - Direct LDAP domain lookups, 34-29
- ldaps: URLs
 - MTA URL types, 1-4
- ldapsearchtimeout base option, 7-10
 - Direct LDAP alias lookups, 35-5
 - Direct LDAP domain lookups, 34-29
- ldaptimeout option (DEPRECATED), 27-17
- ldaptrace base option, 7-10
- ldapurl MMP/IMAP Proxy/POP Proxy/SUBMIT
Proxy option, 27-17
- ldapusessl PAB option, 59-2
- ldap_add_header MTA option, 39-139
- ldap_alternate_recipient MTA option, 39-122
- ldap_attr_domain1_schema2 MTA option, 39-81,
39-142
- ldap_attr_domain2_schema2 MTA option, 39-82,
39-142
- ldap_attr_domain_search_filter MTA option,
39-82, 39-88, 39-142
- ldap_auth_attr_mail_host MTA option, 39-151
- ldap_auth_attr_recall_secret MTA option, 39-152
- ldap_auth_attr_sender MTA option, 39-151
- ldap_auth_attr_submit_channel MTA option,
39-151
 - Use with FUTURERELEASE, 49-12
- ldap_auth_domain MTA option, 39-133
- ldap_auth_mappingN MTA option, 39-141
- ldap_auth_password MTA option, 39-133
- ldap_auth_policy MTA option, 39-132
- ldap_auth_url MTA option, 39-132
- ldap_autoreply_addresses MTA option
 - Vacation message not generated, 5-45
- ldap_autoreply_reply MTA option
 - Vacation message not generated, 5-45
- ldap_autoreply_text MTA option
 - Vacation message not generated, 5-46

ldap_autoreply_text_internal MTA option
 Vacation message not generated, 5–46
 vnd.sun.autoreply-internal Sieve environment item, 5–15

ldap_autoreply_timeout MTA option, 39–65

ldap_autosecretary MTA option, 39–122

ldap_basedn_filter_schema1 base option, 7–7, 39–82, 39–89

ldap_basedn_filter_schema1 MTA option, 7–7, 39–82, 39–89

ldap_basedn_filter_schema2 base option, 7–7, 39–82, 39–89

ldap_basedn_filter_schema2 MTA option, 7–7, 39–82, 39–89

ldap_blocklimit MTA option, 33–112, 39–123

ldap_cant_domain MTA option, 39–132

ldap_cant_url MTA option, 39–132

ldap_capture MTA option, 39–116

ldap_check_header MTA option, 39–141

ldap_conversion_tag MTA option, 39–123

ldap_creation_date MTA option, 39–151

ldap_default_attr MTA option, 39–86

ldap_default_domain MTA option, 39–82, 39–97
 Direct LDAP alias lookups, 35–6
 Direct LDAP domain lookups, 34–30
 Twin of base.defaultdomain, 7–5, 27–13

ldap_delay_notifications MTA option, 39–138

ldap_delivery_file MTA option, 39–125

ldap_delivery_option MTA option, 39–120
 Deferred expansion of groups, 39–184
 delivery_options interpretation, 39–92

ldap_detourhost_optin MTA option, 39–123

ldap_disk_quota MTA option, 39–124
 User LDAP attribute to override defaultmailboxquota, 16–11

ldap_domain_attr_alias base option, 7–6, 39–143

ldap_domain_attr_alias MTA option, 7–6, 39–143

ldap_domain_attr_autoreply_timeout MTA option, 39–65, 39–147

ldap_domain_attr_autosecretary MTA option, 39–146

ldap_domain_attr_basedn base option, 7–6, 39–143

ldap_domain_attr_basedn MTA option, 7–6, 39–143

ldap_domain_attr_blocklimit MTA option, 33–112

ldap_domain_attr_capture MTA option, 39–148

ldap_domain_attr_catchall_address MTA option, 39–149

ldap_domain_attr_catchall_mapping MTA option, 39–149

ldap_domain_attr_conversion_tag MTA option, 39–146

ldap_domain_attr_creation_date MTA option, 39–151

ldap_domain_attr_detourhostoptin MTA option, 39–150

ldap_domain_attr_disk_quota MTA option, 39–147

ldap_domain_attr_filter MTA option, 39–148
 Sieve hierarchy, 5–65

ldap_domain_attr_mailserv MTA option, 39–143

ldap_domain_attr_mail_status base option, 7–7, 39–145

ldap_domain_attr_mail_status MTA option, 7–7, 39–145
 Hold channel, 52–10
 Releasing messages, 52–11

ldap_domain_attr_message_quota MTA option, 39–148

ldap_domain_attr_nosolicit MTA option, 39–147

ldap_domain_attr_presence MTA option, 39–146

ldap_domain_attr_recipientcutoff MTA option, 33–88, 33–121, 39–150

ldap_domain_attr_recipientlimit MTA option, 33–88, 33–121, 39–150

ldap_domain_attr_report_address MTA option, 39–148

ldap_domain_attr_routing_hosts MTA option, 39–144
 Routing to a gateway system, 49–47

ldap_domain_attr_smarthost MTA option, 39–144
 Routing to a gateway system, 49–47

ldap_domain_attr_sourceblocklimit MTA option, 33–112, 39–149

ldap_domain_attr_source_channel MTA option, 33–82, 39–150
 userswitchchannel channel option, 33–20

ldap_domain_attr_source_conversion_tag MTA option, 39–146

ldap_domain_attr_status base option, 7–7, 39–145

ldap_domain_attr_status MTA option, 7–7, 39–145

ldap_domain_attr_subaddress MTA option, 39–144
 Subaddresses and LDAP lookups, 35–45

ldap_domain_attr_uid_separator base option, 7–6, 39–144

ldap_domain_attr_uplevel MTA option, 39–143

ldap_domain_filter_schema1 base option, 7–8, 39–83, 39–89
 Direct LDAP domain lookups, 34–30

ldap_domain_filter_schema1 MTA option, 7–8, 39–83, 39–89
 Direct LDAP domain lookups, 34–30

ldap_domain_filter_schema2 base option, 7–8, 39–83, 39–89
 Direct LDAP domain lookups, 34–30

ldap_domain_filter_schema2 MTA option, 7–8, 39–83, 39–89
 Direct LDAP domain lookups, 34–30

`ldap_domain_known_attributes` base option, 7–5, 39–83
 Direct LDAP domain lookups, 34–30
`ldap_domain_known_attributes` MTA option, 7–5, 39–83
 Direct LDAP domain lookups, 34–30, 34–30
`ldap_domain_root` MTA option, 39–83, 39–89
 Direct LDAP alias lookups, 35–6
 Direct LDAP domain lookups, 34–30
 Twin of `base.dn`, 7–16
`ldap_domain_timeout` base option, 7–5, 39–83, 39–153
`ldap_domain_timeout` base/MTA option
 TCP wrappers, 6–2
`ldap_domain_timeout` MTA option, 7–5, 39–83, 39–153
 Direct LDAP domain lookups, 34–30, 34–30
`ldap_end_date` MTA option, 39–123
 Current date comparison for vacation message generation, 5–45
`ldap_errors_to` MTA option, 39–137
`ldap_expandable` MTA option
 `expn*` channel options, 33–126
`ldap_filter` MTA option, 39–129
 Sieve hierarchy, 5–65
`ldap_filter_reference` MTA option
 Sieve hierarchy, 5–65
`ldap_global_config_templates` MTA option, 39–89
`ldap_group_dn` MTA option, 39–135
`ldap_group_mail_status` MTA option, 39–115
`ldap_group_object_classes` MTA option, 39–89
`ldap_group_object_classes` MTA option
 Direct LDAP alias lookups, 35–6
`ldap_group_rfc822` MTA option, 39–137
`ldap_group_status` MTA option, 39–115
`ldap_hoh_filter` MTA option, 39–97, 39–141
`ldap_hoh_owner` MTA option, 39–97, 39–141
 Sieve syntax error notification messages, 47–2
`ldap_host` MTA option, 39–76
 Direct LDAP alias lookups, 35–5
 Direct LDAP domain lookups, 34–29
 Twin of `ugldaphost` base option, 7–14
`ldap_host_alias_list` base option, 7–9
`ldap_host_alias_list` MTA option, 39–84, 39–98
 Direct LDAP alias lookups, 35–6
`ldap_jettison_domain` MTA option, 39–131
`ldap_jettison_url` MTA option, 39–131
`ldap_list_advertised` MTA option, 39–187
`ldap_list_description` MTA option, 39–187
`ldap_list_name` MTA option, 39–187
`ldap_list_public_roster` MTA option, 39–187
`ldap_list_subscribe_policy` MTA option, 39–187
`ldap_list_trust_new_members` MTA option, 39–187
`ldap_list_unsubscribe_policy` MTA option, 39–187
`ldap_local_host` MTA option, 39–84, 39–98
 Direct LDAP alias lookups, 35–6
 L channel `official_host_name`, 33–79
 Twin of `base.hostname`, 7–9
`ldap_mailhost` MTA option, 39–124
`ldap_mail_aliases` MTA option, 39–87
 Direct LDAP alias lookups, 35–6
`ldap_mail_reverses` MTA option, 39–87
`ldap_maximum_message_size` MTA option, 33–112, 39–133
`ldap_max_connections` MTA option, 39–76
 Direct LDAP domain lookups, 34–30, 35–6
`ldap_message_quota` MTA option, 39–125, 39–125
 User LDAP attribute to override
 `defaultmessagequota`, 16–11
`ldap_mlsrange` MTA option, 39–116
`ldap_mls_sub_action_key` MTA option, 39–187
`ldap_mls_sub_digest` MTA option, 39–187
`ldap_mls_sub_join_date` MTA option, 39–187
`ldap_mls_sub_join_ip` MTA option, 39–187
`ldap_mls_sub_list_id` MTA option, 39–187
`ldap_mls_sub_mail` MTA option, 39–187
`ldap_mls_sub_object_class` MTA option, 39–186
`ldap_mls_sub_receive_mail` MTA option, 39–187
`ldap_mls_sub_role` MTA option, 39–187
`ldap_mls_sub_suppress_duplicates` MTA option, 39–187
`ldap_mls_sub_tentative_email` MTA option, 39–187
`ldap_mls_sub_track` MTA option, 39–187
`ldap_mluser_basedn` MTA option, 39–186
`ldap_mluser_join_date` MTA option, 39–186
`ldap_mluser_join_ip` MTA option, 39–186
`ldap_mluser_mail` MTA option, 39–186
`ldap_mluser_name` MTA option, 39–186
`ldap_mluser_object_class` MTA option, 39–186
`ldap_mluser_password` MTA option, 39–186
`ldap_mluser_unique_id` MTA option, 39–186
`ldap_moderator_url` MTA option, 39–134
`ldap_optin*` MTA options, 39–121
`ldap_pab_host` MTA option, 39–183
`ldap_pab_max_connections` MTA option, 39–183
`ldap_pab_password` MTA option, 39–183
`ldap_pab_port` MTA option, 39–183
`ldap_pab_username` MTA option, 39–183
`ldap_password` MTA option, 39–76
 Direct LDAP alias lookups, 35–5
 Direct LDAP domain lookups, 34–29
 Twin of `base.ugldapbindcred`, 7–14
`ldap_personal_name` MTA option, 39–120
 PERSONAL_NAMES mapping table, 35–55
`ldap_port` MTA option, 39–77
 Direct LDAP alias lookups, 35–5

- Direct LDAP domain lookups, 34–29
- Twin of `ugldapport` base option, 7–15
- `ldap_presence` MTA option, 39–122
- `ldap_primary_address` MTA option, 39–120
- `ldap_program_info` MTA option
 - \$P substitution in LDAP URLs, 1–7
- `ldap_recipientcutoff` MTA option, 33–88, 33–121, 39–117
- `ldap_recipientlimit` MTA option, 33–88, 33–121, 39–117
- `ldap_reject_action` MTA option, 39–131
- `ldap_reject_text` MTA option, 39–131
- `ldap_remove_header` MTA option, 39–139
- `ldap_reprocess` MTA option, 39–130
 - Deferred expansion of groups, 39–184
 - Mass mailings, 36–21
- `ldap_routing_address` MTA option, 39–119
- `ldap_schemalevel` base option, 7–5, 39–90
- `ldap_schemalevel` MTA option, 7–5, 39–90
- `ldap_schematag` MTA option, 39–90
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
- `ldap_sourceblocklimit` MTA option, 33–112, 39–118
- `ldap_source_channel` MTA option, 39–118
 - Name of attribute used for `userswitchchannel` purposes, 33–82
 - `userswitchchannel` channel option, 33–20
- `ldap_source_conversion_tag` MTA option, 39–120
- `ldap_source_optin*` MTA options, 39–118
 - Archiving, 54–21
- `ldap_spare_4` MTA option
 - SIEVE_EXTLISTS mapping probes, 5–30
- `ldap_spare_5` MTA option
 - SIEVE_EXTLISTS mapping probes, 5–30
- `ldap_spare_6` MTA option
 - SIEVE_EXTLISTS mapping probes, 5–30
- `ldap_start_date` MTA option, 39–122
 - Current date comparison for vacation message generation, 5–45
- `ldap_timeout` MTA option, 39–77
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- `ldap_uid` MTA option, 39–115
 - \$M substitution in LDAP URLs, 1–7
- `ldap_uid_invalid_chars` MTA option, 39–99
- `ldap_url_result_mapping` MTA option, 39–137
 - Example, 36–13
- `ldap_username` MTA option, 39–78
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - Twin of `base.ugldapbinddn`, 7–14
- `ldap_user_mail_status` MTA option, 39–114
 - Hold channel, 52–10
 - Releasing messages, 52–11
- `ldap_user_object_classes` MTA option, 39–90
- `ldap_user_object_classeses` MTA option
 - Direct LDAP alias lookups, 35–6
- `ldap_user_root` MTA option, 39–87, 39–91
 - Direct LDAP alias lookups, 35–6
 - Twin of `base.ugldapbasedn`, 7–14
- `ldap_user_status` MTA option, 39–114
- `ldap_use_async` MTA option, 39–77
 - Mass mailings, 36–21
- Legacy configuration
 - `option.dat` file, 39–7, 39–9
- `legacy_proxyauth` IMAP option, 21–3
- Length limits in configuration, 39–86
 - Alias file
 - Alias length, 35–24
 - Alias translation address length, 35–24
 - Number of alias translation addresses, 35–24
 - Physical line, 35–24
 - `alias_urlN` template
 - LDAP URL substitution results, 35–7
 - Channel names, 33–2
 - Channel option arguments, 33–7
 - Conversion tags, 38–2
 - `delivery_options` MTA option
 - Length of each clause, 39–95
 - Number of clauses, 39–92, 39–95
 - Header label length
 - 256 characters, `addheader` Sieve action, 5–25
 - `mailAutoReplySubject` value, 39–127
 - Mapping tables, 37–3
 - Probe strings, 38–3
 - Message size
 - SpamAssassin, `MESSAGE_BUFFER_SIZE` SpamAssassin option, 45–7
 - Option value, 39–10
 - `reverse_url` MTA option value, 39–88
 - Rewrite rules
 - LDAP URL substitution results, 34–23
 - Pattern length, 34–2
 - Template length, 34–2
 - Sieve filter string length when variables are enabled, 39–227
 - `spamfilterN_action_M` values, 39–237
 - `spamfilterN_string_action`, 39–239
 - `spamfilterN_verdict_M` values, 39–236
- `limitheadertermination` channel option, 33–73
- Line continuation
 - In aliases file, 35–24
- Line wrapping
 - For display
 - CHARSET-CONVERSION mapping table, 38–17

- CONVERSIONS mapping table, 38–4
 - Header lines
 - LINELENGTH header trimming option, 33–157
 - MIME parameter segmentation, 33–50, 33–54
 - linelength channel option, 33–47
 - linelimit channel option, 33–112
 - error_text_line_over MTA option, 39–159
 - lines_to_return MTA option, 39–214
 - line_limit MTA option, 33–112, 39–207
 - Linux
 - dbtmpdir Message Store option, 16–11
 - lockdir base option, 7–10
 - tmpdir base option, 7–14
 - tmpdir Message Store archive option, 16–14
 - tmpdir MTA option, 39–154
 - listenaddr base option, 7–17
 - ENS server host, 61–1
 - listenaddr Dispatcher option, 41–4
 - listenaddr Dispatcher service option, 41–5
 - listenaddr Job Controller option, 42–10
 - listenaddr MeterMaid option, 46–5
 - listenaddr SNMP option, 60–1
 - listen_addresses SMS smpp_relay option, 53–9
 - listen_addresses SMS smpp_server option, 53–12
 - listen_addresses tcp_listen option, 28–1
 - listen_receive_timeout smpp_relay option, 53–9, 53–13
 - listen_receive_timeout smpp_server option, 53–9, 53–13
 - listen_transmit_timeout SMS smpp_relay and smpp_server option, 53–9, 53–13
 - listimplicit Message Store option, 16–6
 - listrecover Message Store deleted option, 16–26
 - listurl base option, 7–8
 - lmtmp channel option, 33–127
 - LMTP channels, 49–13
 - Client, 49–13
 - defragment option, 52–3
 - delivery_options MTA option, 39–94
 - multigate channel option, 39–95
 - Rewrite rules, 39–95
 - Line terminators
 - lmtmp* channel options, 33–128
 - Message Store stress, 42–4
 - 250 2.3.99 Delivery OK but store under stress, 42–4
 - noticehost alarm option, 11–1
 - Options, 49–16
 - See also TCP/IP channels, Options, 49–16
 - See also TCP/IP channels, 49–2
 - Server, 49–14
 - BUFFER_SIZE TCP/IP-channel-specific option, 49–22
 - In-memory buffering of incoming messages, 49–22
 - Line terminator, 33–128
 - Line terminator(s), 33–127
 - lmtmp* channel options, 33–127
 - msprobe probe of, 10–2
 - Options, 49–15
 - LMTP commands
 - LHLO
 - \$L input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148
 - BANNER_HOST TCP/IP-channel-specific option, 49–21
 - BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 - Host name, 33–80
 - RCPT TO
 - XAFLG parameter, 33–109, 33–124
 - XDFLG parameter, 33–109, 33–124
 - See also SMTP commands, 49–13
 - lmtmp* channel options
 - Imply nonotary, 33–96, 33–131
 - lmtmp_cr channel option, 33–127
 - lmtmp_crlf channel option, 33–127
 - lmtmp_crorlf channel option, 33–127
 - lmtmp_lf channel option, 33–127
 - Local channel, 52–1
 - Removal of source routes during rewriting, 34–8
 - Source routes
 - Removal during rewriting, 34–8
 - local.hostname configutil parameter, 39–84, 39–98
 - local.imta.hostnamealiases configutil parameter
 - MTA use, 39–84, 39–98
 - localbehavior channel option, 33–39
 - Localization
 - Welcome message for new Message Store users
 - welcomemsg message_language option, 17–1
 - localvrfy channel option, 33–125
 - local_format_restrictions MTA option, 39–57
 - local_host_alias channel option, 33–80
 - Overridden by BANNER_HOST, 49–21
 - Overridden by BANNER_REVERSE_HOST, 49–21
 - local_quota_checks MTA option
 - RESTRICTED, 39–208
 - lockdir base option, 7–10
 - lockmailbox POP option, 22–2
 - Locks
 - BDB
 - dblockcount base option, 7–10
 - logauthsessionid IMAP option, 21–11

logcommands IMAP option, 21–4
 logdir logfile option, 27–17
 logexpungedetails Message Store option, 16–6
 logfile options, 7–19
 expirytime, 7–19
 filemode, 7–19
 flushinterval, 7–19
 logdir, 27–17
 logmillisecond, 7–19
 maxlogfiles, 7–19
 maxlogfilesize, 7–20
 maxlogsize, 7–20
 rolloverpolicy, 7–20
 rollovertime, 7–20
 syslogfacility, 7–21
 logfilename Dispatcher service option, 41–5
 Logging
 default file
 Errors, msprobe timeouts, 10–1
 Warnings, msprobe timeouts, 10–2
 IMAP
 logunauthsession IMAP option, 21–13
 imapcmd file, 21–4
 ims-ms channels
 LOG_DEQUEUE_RATE ims-ms-channel-specific option, 51–7
 imta file
 ims-ms channel debugging, 51–7
 LMTP server
 logfilename Dispatcher option, 41–5
 logcommands IMAP option, 21–4
 logexpungedetails Message Store option, 16–6
 maxlog Message Store option, 16–6
 maxlogfiles option, 7–20
 Message Store transaction
 actionattributes option, 21–11, 23–1
 actions option, 21–11, 23–1
 msgtrace file
 activate Message Trace option, 23–1
 ims-ms channel, 51–7
 MSHTTP
 logunauthsession MSHTTP option, 29–13
 MTA log files
 Purging of, 8–3, 16–24
 MTA message return job
 return_split_period MTA option, 39–270
 return_verify MTA option, 39–75
 MTA transaction, 55–1
 Application information, Syntax of, 55–8
 B records, MAX_B_ENTRIES TCP/IP-channel-specific option, 49–29
 Bitbucket channel, 52–2
 BURL use, 49–11
 Connections, C entries, 44–4
 Connections, T entries, 44–4
 Connections, T records, 44–3
 Format, 55–2
 Format, ETRN client host name, 39–264
 Format, J records corresponding to rejections due to message size, 33–113
 Format, logheader channel option, 33–84
 Format, log_auth MTA option, 39–252
 Format, log_callout_delays MTA option, 39–252
 Format, log_connection MTA option, 39–254
 Format, LOG_CONNECTION TCP/IP-channel-specific option, 49–27
 Format, log_conversion_tag MTA option, 39–255
 Format, log_delivery_flags MTA option, 39–262
 Format, log_diagnostics MTA option, 39–256
 Format, log_envelope_id MTA option, 39–256
 Format, log_filename MTA option, 39–256
 Format, log_filter MTA option, 39–231, 39–257
 Format, log_format MTA option, 39–257
 Format, log_futurerelease MTA option, 39–261
 Format, log_header MTA option, 39–261
 Format, log_imap_flags MTA option, 39–262
 Format, log_intermediate MTA option, 39–263
 Format, log_local MTA option, 39–263
 Format, log_mailbox_uid MTA option, 39–263
 Format, log_message_id MTA option, 39–263
 Format, log_mtpriority MTA option, 39–264
 Format, log_node MTA option, 39–264
 Format, log_notary MTA option, 39–265
 Format, log_priority MTA option, 39–265
 Format, log_process MTA option, 39–266
 Format, log_queue_time MTA option, 39–266
 Format, log_reason MTA option, 39–267
 Format, log_sensitivity MTA option, 39–267
 Format, log_times MTA option, 39–268
 Format, log_tracking MTA option, 39–268
 Format, log_transactionlog MTA option, 39–232, 39–268
 Format, log_uid MTA option, 39–269
 Format, log_username MTA option, 39–269
 Format, log_use_xtext MTA option, 39–269
 Format, SASL error, 39–264
 Format, XML compatible, 39–258
 J records, MAX_J_ENTRIES TCP/IP-channel-specific option, 49–30
 J records, Message size restrictions, 33–113
 J records, SPF HELP/EHLO check failure, 33–144

logging channel option, 33–84, 33–84
 logheader channel option, 33–84
 log_alternate_recipient MTA option, 39–252
 log_auth MTA option, 39–252
 LOG_BANNER TCP/IP-channels-specific option, 49–27
 log_callout_delays MTA option, 39–252
 log_connection MTA option, 39–254
 LOG_CONNECTION TCP/IP-channel-specific option, 49–27
 log_connections_syslog MTA option, 39–247
 log_deliver_by MTA option, 39–255
 log_diagnostics MTA option, 39–256
 log_envelope_id MTA option, 39–256
 log_filename MTA option, 39–256
 log_filter MTA option, 39–231, 39–257
 log_format MTA option, 39–257
 log_futurerelease MTA option, 39–261
 log_header MTA option, 39–261
 log_imap_flags MTA option, 39–262
 log_intermediate MTA option, 39–262
 log_local MTA option, 39–263
 log_mailbox_uid MTA option, 39–263
 log_messages_syslog MTA option, 39–247
 log_message_id MTA option, 39–263
 log_mtpriority MTA option, 39–264
 log_node MTA option, 39–264
 log_notary MTA option, 39–265
 log_priority MTA option, 39–265
 log_process MTA option, 39–266
 log_queue_time MTA option, 39–266
 log_reason MTA option, 39–267
 log_sensitivity MTA option, 39–267
 log_sndopr MTA option, 39–249
 log_times MTA option, 39–268
 log_tracking MTA option, 39–268
 log_transactionlog MTA option, 39–232, 39–268
 LOG_TRANSPORTINFO TCP/IP-channel-specific option, 49–28
 log_uid MTA option, 39–269
 log_username MTA option, 39–269
 log_use_xtext MTA option, 39–269
 Message size, Reported in units of MTA blocks, 39–206
 Q records, Too many failures to this host during this run; skipping this host:, 33–135
 separate_connection_log MTA option, 39–270
 Size of message, Reported in units of MTA blocks, 39–206
 transactionlog Sieve action, 5–18
 Transport information, Syntax of, 55–7
 X record, SMTP disconnect, 49–20

nslog
 Rollover, rolloverpolicy, 7–20
 Rollover, maxlogfilesize option, 7–20
 Rollover, maxlogsize option, 7–20
 Rollover, rollovertime option, 7–20
 NT event log
 Notices generated by address access mapping tables, 44–9
 POP
 logunauthsession POP option, 22–4
 poplogmboxstat POP option, 22–2
 return_job
 return_verify MTA option, 39–75
 rollovermanager, 15–1
 S/MIME applet
 appletlogging S/MIME option, 30–7
 See also logfile options, 7–19
 SMTP server
 logfilefilename Dispatcher option, 41–5
 SUBMIT server
 logfilefilename Dispatcher option, 41–5
 syslog
 Line length maximum, 39–247, 39–249
 log_connections_syslog MTA option, 39–247
 log_messages_syslog MTA option, 39–247
 log_sndopr MTA option, 39–249
 MTA options, 39–246
 Notices generated by address access mapping tables, 44–9
 sndopr_priority MTA option, 39–249
 syslogfacility logfile option, 7–21
 Telemetry
 forcetelemetry icapservice option, 32–1
 forcetelemetry IMAP option, 21–11
 forcetelemetry MSHTTP option, 29–12
 forcetelemetry POP option, 22–3
 Less private than imap.logcommands output, 21–4
 X record
 SMTP disconnect, 49–19
 logging channel option, 33–84, 55–1
 logheader channel option, 33–84
 Logical name table (OpenVMS)
 name_table_name MTA option, 39–59
 loginpw smime option, 30–2
 loginseparator base option, 7–16
 loglevel ENS option, 39–72, 61–2
 loglevel imaproxy option, 27–18, 39–72
 loglevel messagetrace option, 23–1, 39–72
 loglevel MMP option, 27–17, 39–72
 loglevel MTA option, 39–72, 49–16
 loglevel option, 39–72
 ims-ms channels, 51–6

- loglevel popproxy option, 27–18, 39–72
 - loglevel submitproxy option, 27–18, 39–73
 - logmillisecond logfile option, 7–19
 - logprotocolerrors IMAP option, 21–11
 - logprotocolerrors POP option, 22–3
 - logunauthsession IMAP option, 21–13
 - logunauthsession MSHTTP option, 29–13
 - logunauthsession POP option, 22–4
 - loguser notifytarget option, 24–4
 - log_alq MTA option, 39–173, 39–251
 - log_alternate_recipient MTA option, 39–252
 - log_connection MTA option, 39–254
 - \$T flag in PORT_ACCESS mapping table, 44–4
 - Example, 55–4
 - log_connections_syslog MTA option, 39–247
 - log_conversion_tag MTA option, 39–255
 - log_debug MTA option, 39–74
 - log_delay_bins MTA option, 39–70
 - log_delivery_flags MTA option, 39–262
 - log_deliver_by MTA option, 39–255
 - log_deq MTA option, 39–173, 39–251
 - log_diagnostics MTA option, 39–256
 - log_envelope_id MTA option, 39–256
 - Example, 55–4
 - log_filename MTA option, 39–256
 - Example, 55–4
 - log_filter MTA option, 39–231, 39–257
 - addprefix or addsuffix actions, 5–49
 - Diagnosing .HELD files, 52–11
 - discard or jettison strings, 5–23
 - Example, 55–4
 - Memcache protocol errors, 5–63
 - Sieve duplicate errors, 5–25, 5–63
 - Sieve vacation errors, 5–44, 5–46, 5–63
 - spamttest level and virustest level, 5–42
 - systemfilter MTA option, 39–223
 - vacation action, 5–44
 - log_format MTA option, 39–257
 - log_frustration_limit MTA option, 39–70
 - log_futurerelease MTA option, 39–261
 - log_header MTA option, 39–261
 - Affected by log_messages_syslog, 39–249
 - Compared with transactionlog use in Sieve script, 39–232, 39–269
 - log_imap_flags MTA option, 39–262
 - log_intermediate MTA option, 39–263
 - Example, 55–4
 - log_local MTA option, 39–263
 - log_messages_syslog MTA option, 39–247
 - log_message_id MTA option, 39–263
 - Example, 55–4
 - log_node MTA option, 39–264
 - Example, 55–4
 - log_notary MTA option, 39–265
 - Example, 55–4
 - log_priority MTA option, 39–265
 - Example, 55–4
 - log_process MTA option, 39–266
 - Example, 55–4
 - Reprocess channel, 52–19
 - log_queue_time MTA option, 39–266
 - log_reason MTA option, 39–267
 - log_sensitivity MTA option, 39–267
 - Example, 55–4
 - log_size_bins MTA option, 39–70
 - log_sndopr MTA option, 39–249
 - log_statistics MTA option, 39–70
 - log_tracking MTA option, 39–268
 - log_transactionlog MTA option, 39–232, 39–268
 - log_username MTA option, 39–269
 - Example, 55–4
 - filter_discard channel logs as FILTER_DISCARD, 52–9
 - log_use_xtext MTA option, 39–269
 - Loop
 - CPU
 - Alias nesting limit, max_alias_levels MTA option, 39–58
 - Alias recursion limit, 35–46
 - Mapping table processing iteration limit, 37–8
 - Rewrite rule repeated rewriting, 34–15
 - Sieve filter loop construct, 5–53
 - Message
 - See Looping message, 52–11
 - Message routing
 - loopcheck channel option, 33–128
 - Received: header line MTA options, 39–220
 - Notification messages
 - returnenvelope channel option, 33–99
 - return_address MTA option, 39–215
 - Vacation messages, RFC 3834, 5–46
 - loopcheck channel option, 33–128
 - Looping message
 - Troubleshooting
 - .HELD files, 52–11
- ## M
- MacMIME
 - Format conversions, 38–21
 - See also RFC 1740, 38–21
 - Mail filtering, 44–1
 - Sieve filters, 5–1, 44–1
 - Sieve language, 5–2
 - mailboxexpungesize Message Store deleted option, 16–27
 - mailboxpurgedelay Message Store option, 16–12

mailfromdnsverify channel option, 33–129, 33–140

- Bit in returnenvelope, 33–99
- Bit in return_envelope, 39–156, 39–215
- DNS verification
 - test -rewrite utility, 58–35
- error_text_mailfromdnsverify MTA option, 39–165

mailhostattrs mmp/imaproxy/popproxy/vdomain option, 27–18

Mailing list and group MTA options, 39–184

Mailing lists, 36–1

- Access control for expansion
 - DISABLE_EXPAND TCP/IP-channel-specific option, 49–24
 - expandable_default MTA option, 39–184
 - expn* channel options, 33–126
 - Makes use of access control for postings, 39–185
- Access control for postings, 36–19
 - alias_and alias option, 35–10
 - alias_auth_channel alias option, 35–10
 - alias_auth_list alias option, 35–10
 - alias_auth_mapping alias option, 35–10
 - alias_auth_username alias option, 35–10
 - alias_cant_channel alias option, 35–10
 - alias_cant_mapping alias option, 35–10
 - alias_cant_username alias option, 35–10
 - alias_moderator_address, 35–17
 - alias_moderator_list, 35–17
 - alias_moderator_mapping, 35–17
 - alias_or alias option, 35–10
 - alias_username_moderator_list, 35–17
 - AND alias file named parameter, 35–27
 - AUTH_CHANNEL alias file named parameter, 35–27
 - AUTH_LIST alias file named parameter, 35–27
 - AUTH_MAPPING alias file named parameter, 35–28
 - AUTH_USERNAME alias file named parameter, 35–29
 - CANT_CHANNEL alias file named parameter, 35–27
 - CANT_LIST alias file named parameter, 35–27
 - CANT_MAPPING alias file named parameter, 35–28
 - CANT_USERNAME alias file named parameter, 35–29
 - Deferred expansion interactions, 36–19
 - Example, 36–20
 - Interpretation of multiple, 36–2
 - Moderator of non-member attempted postings, 36–20
 - MODERATOR_ADDRESS, 35–35
 - MODERATOR_LIST, 35–35
 - MODERATOR_MAPPING, 35–35
 - OR alias file named parameter, 35–27
 - Password, 36–3
 - PASSWORD alias file named parameter, 35–37
 - Password, alias_password alias option, 35–20
 - Password, Example, 36–20
 - SMTP AUTH use required, 36–20, 36–20
 - USERNAME_AUTH_LIST alias file named parameter, 35–27
 - USERNAME_CANT_LIST alias file named parameter, 35–27
 - USERNAME_MODERATOR_LIST, 35–35
- Addresses of, 36–1
- Alias options
 - alias_hold_*, 35–16
 - alias_moderator_address, 35–17
 - alias_moderator_list, 35–17
 - alias_moderator_mapping, 35–17
 - alias_username_moderator_list, 35–17
- Attachments
 - alias_prefix_text alias option, 35–20
 - alias_suffix_text alias option, 35–20
- Constructing list member addresses, 36–13
- Deferred expansion
 - defer_group_processing MTA option, 39–184
 - Members vs. access controls, 36–19
- Delivery receipt request
 - SMTP response to RCPT TO, error_text_receipt_it MTA option, 39–162
- Digests
 - digest_on MTA option, 39–184
- Duplicate message elimination
 - alias_header_check alias option, 35–16
 - Copies to members of multiple sub-lists, 36–19
 - HEADER_CHECK named parameter, 35–35
 - ldap_check_header MTA option, 39–141
 - See also Message, Duplicate, 35–35
- Dynamic
 - Example, 36–10
- Envelope From address
 - alias_envelope_from alias option, 35–14
- Errors
 - alias_error_text alias option, 35–14
- Example
 - Disallow replies to prior postings, 36–5
- Forwarding

- ldap_forwarding_address MTA option, 39–130
- Head-of-household controls
 - ldap_filter_reference MTA option, 39–129
 - ldap_parental_controls MTA option, 39–129
- Header
 - Approved:, 36–3
 - Approved:, alias_password alias option, 35–20
 - Approved:, PASSWORD named parameter, 35–37
 - Deferred-delivery:, DEFERRED named parameter, 35–31
 - Deferred-delivery:, DEFERRED_LIST named parameter, 35–31
 - Deferred-delivery:, DEFERRED_MAPPING named parameter, 35–31
 - Expiry-date:, 35–33
 - HEADER_ADDITION alias file named parameter, 35–34
 - Received:, 35–38
 - Subject:, Tag on postings, 35–39
 - To:, TO named parameter, 35–40
- LDAP attributes
 - Alternate to uniqueMember, 39–136
 - Auto-secretary use, 39–122
 - Capture trigger, 39–116
 - expandable, 39–140
 - Group status, 39–115
 - GROUP_AUTH mapping table, 36–21
 - inetMailGroupStatus, 39–115
 - ldap_group_last_access_time MTA option, 39–134
 - ldap_group_url1 MTA option, 39–135
 - ldap_group_url2 MTA option, 39–135
 - mail, 39–120
 - mail, Fetched during head of household Sieve filter lookups, 39–130
 - mailAlternateAddress, 39–121
 - mailConversionTag, 39–123
 - mailDeliveryFile, 39–125
 - mailDeliveryFileURL, 39–125
 - mailDeliveryOption, 39–120
 - mailEquivalentAddress, 39–121
 - mailHost, 39–124
 - mailHost,
 - ldap_domain_attr_default_mailhost MTA option, 39–147
 - mailMsgMaxBlocks, 39–123
 - mailProgramDeliveryInfo, 39–125
 - mailRoutingAddress, 39–119
 - mailSieveRuleSource, Fetched during head of household Sieve filter lookups, 39–130
 - mgmanMemberVisibility, 39–140
 - mgrpAddHeader, 39–139
 - mgrpAuthPolicy, 39–133
 - mgrpDelayNotifications, 39–138, 39–138
 - mgrpDigestInterval, 39–138
 - mgrpErrorsTo, 39–137
 - mgrpListTag, 39–139
 - mgrpMaxMessagesPerDay, 39–133
 - mgrpModerator, 39–134
 - mgrpMsgMaxSize, 39–133
 - mgrpMsgPrefixText, 39–140
 - mgrpMsgSuffixText, 39–140
 - mgrpRemoveHeader, 39–139
 - mgrpRFC822MailMember, 39–137
 - mgrpUniqueId, 39–131
 - MLS range, 39–116
 - NO-SOLICITING values, 39–119
 - objectClass, 39–114
 - Opt-in to detour routing, 39–123
 - Opt-in to spam package N, 39–121
 - Personal name, 39–120
 - Preferred country labelling, 39–119
 - preferredLanguage, 39–118
 - Recipient cutoff, 39–117
 - Recipient limit, 39–117
 - rfc822mailalias, 39–121
 - rfc822MailMember, 39–137
 - Source block limit, 39–118
 - Source channel switch, 39–118
 - Source conversion tag, 39–120
 - Source opt-in to spam package N, 39–118
 - Spare N attribute, 39–125
 - Subject: tag, 39–139
 - uid, 39–115
 - uniqueMember, 39–135
 - URL result mapping table, 39–137
 - vacationEndDate, 39–123
 - vacationStartDate, 39–122
- Lists vs. groups, 36–16
- Mass mailings, 36–8
 - defer_group_processing MTA option, 36–21
 - ldap_reprocess MTA option, 36–21
 - ldap_use_async, 36–21
 - mailDeferProcessing LDAP attribute, 36–21
 - Membership, 36–9
 - Sender restrictions, 36–19
- Membership
 - Addresses constructed from non-email SMS attributes, 36–13
 - Constructing e-mail addresses from non e-mail address LDAP attribute values, 36–13
 - Defined via separate groups, 36–18
 - Error reporting, 36–11

Example, 39–136
 Examples, 36–9
 GROUP_TEMPLATES mapping table, 36–15
 Indirect definitions, 36–11
 mail LDAP attribute, 36–10
 Meta definitions, 36–14
 Meta-groups, 39–100
 Nested definitions, 36–18
 Reporting syntax errors, 36–17
 Stored in external LDAP directory, 36–14
 Message size limits
 mgrpMsgMaxSize LDAP attribute, 39–133
 Meta-group list definitions, 36–14
 Meta-groups, 36–14, 39–100
 mgrpErrorsTo LDAP attribute
 Critical for definition, 36–16
 Moderated, 36–3
 alias_*, moderator_*, alias options, 35–17
 For non-members, 36–20
 mgrpMsgRejectAction value of toModerator, 39–131
 Posting access controls, 36–2
 Named parameters, 35–26
 AND, 35–27
 AUTH_CHANNEL, 35–27
 AUTH_LIST, 35–27
 AUTH_MAPPING, 35–28
 AUTH_USERNAME, 35–29
 BLOCKLIMIT, 35–30
 BLOCKLIMIT, error_text_list_block_over MTA option, 39–159
 BLOCKLIMIT, error_text_user_block_over MTA option, 39–160
 CANT_CHANNEL, 35–27
 CANT_LIST, 35–27
 CANT_MAPPING, 35–28
 CANT_USERNAME, 35–29
 CAPTURE, 35–30
 CONVERSION_TAG, 35–30
 CREATION_DATE, 35–31
 DEFERRED, 35–31
 DEFERRED_LIST, 35–31
 DEFERRED_MAPPING, 35–31
 DELAY_NOTIFICATIONS, 35–32
 DIGEST_RECURRENCE, 35–32
 DIRECT_LIST, 35–32
 DIRECT_MAPPING, 35–32
 ENVELOPE_FROM, 35–32, 36–16
 ERROR_TEXT, 35–33
 EXPANDABLE, 35–33, 39–185
 EXPIRY, 35–33
 FILTER, 35–33
 HEADER_ADDITION, 35–34
 HEADER_ADDITION, Compared to use of mgrpAddHeader group LDAP attribute, 39–139
 HEADER_ALIAS, 35–34
 HEADER_CHECK, 35–35
 HEADER_EXPANSION, 35–34
 HEADER_TRIM, 35–34
 HEADER_TRIM, Compared to use of mgrpRemoveHeader group LDAP attribute, 39–139
 HOLD_LIST, 35–35
 HOLD_MAPPING, 35–35
 IMPORTANCE, 35–35
 JOURNAL, 35–30
 KEEP_DELIVERY, 35–35
 KEEP_READ, 35–35
 LINELIMIT, 35–30
 LINELIMIT, error_text_list_line_over MTA option, 39–160
 LINELIMIT, error_text_user_line_over MTA option, 39–160
 LIST_NAME, 35–35
 MODERATOR_ADDRESS, 35–35
 MODERATOR_LIST, 35–35
 MODERATOR_MAPPING, 35–35
 NODELAY_NOTIFICATIONS, 35–32
 NOHOLD_LIST, 35–35
 NOHOLD_MAPPING, 35–35
 NONEXPANDABLE, 35–33, 39–185
 NOORIGINATOR_REPLY, 35–37
 NORECEIVEDFOR, 35–38
 NORECEIVEDFROM, 35–38
 NOSOLICIT, 35–36
 OPTIN, 35–37
 OPTIN1, 35–37
 OPTIN2, 35–37
 OPTIN3, 35–37
 OPTIN4, 35–37
 OPTIN5, 35–37
 OPTIN6, 35–37
 OPTIN7, 35–37
 OPTIN8, 35–37
 OR, 35–27
 ORIGINATOR_REPLY, 35–37
 PASSWORD, 35–37
 PRECEDENCE, 35–35
 PREFIX_TEXT, 35–37
 PRIORITY, 35–35
 PRIVATE, 35–38
 PUBLIC, 35–38
 RECEIVEDFOR, 35–38
 RECEIVEDFROM, 35–38
 REPROCESS, 35–38

- SASL_AUTH_LIST, 35–39
- SASL_AUTH_MAPPING, 35–39
- SASL_CANT_LIST, 35–39
- SASL_CANT_MAPPING, 35–39
- SASL_MODERATOR_LIST, 35–39
- SASL_MODERATOR_MAPPING, 35–39
- SENSITIVITY, 35–35
- SEQUENCE_PREFIX, 35–39
- SEQUENCE_STRIP, 35–39
- SEQUENCE_SUFFIX, 35–39
- SINGLE, 35–39
- SPARE*, 35–40
- SUFFIX_TEXT, 35–37
- TAG, 35–40
- TO, 35–40
- USERNAME, 35–40
- USERNAME_AUTH_LIST, 35–27
- USERNAME_CANT_LIST, 35–27
- USERNAME_MODERATOR_LIST, 35–35
- Nested definitions, 36–18
- Notifications, 36–17
 - alias_envelope_from alias option, 35–14
 - Delay notifications, 35–32
 - ldap_delay_notifications MTA option, 39–138
 - ldap_errors_to MTA option, 39–138
 - List owner, 35–14
 - mgrpErrorsTo LDAP attribute, 36–16
 - NOTARY flags, 36–18
- Parental controls
 - ldap_filter_reference MTA option, 39–129
 - ldap_parental_controls MTA option, 39–129
- Password-protected, 36–3
 - password switch for test -rewrite, 58–36
 - reprocessing switch for test -rewrite, 58–37
 - alias_password alias option, 35–20
- Performance tuning, 36–21
- Positional parameters
 - envelope From address, 36–16
- Recursive definition, 35–46
 - max_alias_levels MTA option, 35–46, 39–58
- Sieve filters
 - FILTER named parameter, 35–33
 - ldap_filter MTA option, 39–129
 - ldap_filter_reference MTA option, 39–129
 - Sieve hierarchy, 5–65
- Size limits
 - alias_blocklimit alias option, 35–11
 - alias_linelimit alias option, 35–11
- SMTP EXPN command
 - Checked against access control for postings, 39–185
 - DISABLE_EXPAND TCP/IP-channel-specific option, 49–24
 - expandable LDAP attribute, 39–140
 - expandable_default MTA option, 39–184
 - expn* channel options, 33–126
- Subscription to, 36–22
 - Subaddress, 36–22
- Text additions
 - addprefix and addsuffix Sieve extensions, 5–49
 - alias_prefix_text alias option, 35–20
 - alias_suffix_text alias option, 35–20
 - mgrpMsgPrefixText LDAP attribute, 39–140
 - mgrpMsgSuffixText LDAP attribute, 39–140
- Unique identifier
 - mgrpUniqueId LDAP attribute, 39–131
- Vacation
 - delivery_options MTA option, 39–93
 - ldap_autoreply_addresses MTA option, 39–128
 - ldap_autoreply_mode MTA option, 39–126
 - ldap_autoreply_subject MTA option, 39–126
 - ldap_autoreply_text MTA option, 39–127
 - ldap_autoreply_text_internal MTA option, 39–128
 - ldap_autoreply_timeout MTA option, 39–128
- VERP type functionality, 35–14, 39–138
- MAILSERV
 - LDAP attributes
 - mgrpUniqueId, 39–131
 - LDAP schema
 - MTA options, 39–186
 - List subscriptions
 - LDAP attribute names, 39–187
 - mailserv_moderator_mail MTA option, 39–186
 - mailserv_moderator_uid MTA option, 39–186
 - mailserv_secret MTA option, 39–186
 - Moderator user
 - LDAP attributes, uid, 39–186
 - mailserv_moderator_mail MTA option, 39–186
 - mailserv_moderator_uid MTA option, 39–186
 - MTA options, 39–186
 - MTA options, 39–186
 - LDAP schema, 39–186
 - List LDAP attribute names, 39–187
 - List subscription LDAP attribute names, 39–187
 - mailserv_moderator_mail, 39–186
 - mailserv_moderator_uid, 39–186
 - mailserv_secret, 39–186
 - Moderator user, 39–186
 - User LDAP attribute names, 39–186
 - Unique identifier
 - mgrpUniqueId LDAP attribute, 39–131

-
- Users
 - LDAP attribute names, 39–186
 - MAILSERV lists
 - mgrpBroadcasterPolicy values, 39–132
 - mailserv_moderator_mail MTA option, 39–186
 - mailserv_moderator_uid MTA option, 39–186
 - mailto: URLs
 - Example of mgrpAllowedBroadcaster attribute's value, 36–4
 - Example of mgrpModerator attribute's value, 36–5
 - List-*: header field values, 35–34
 - MTA URL types, 1–4
 - mail_delivery_filename MTA option, 39–272
 - mail_off MTA option, 39–185
 - Mailing list members, 36–22
 - make_source_addresses_unique SMS smpp_relay option, 53–9
 - mapping group, 37–17
 - Mapping tables, 37–1
 - \$C flag
 - Example, 37–20
 - \$E flag
 - Example, 37–20
 - Access control
 - AUTH_ACCESS, 49–37
 - Access mapping tables, 44–2
 - Interaction and timing, 44–15
 - Alias AUTH_MAPPING, 35–28
 - Alias CANT_MAPPING, 35–10, 35–10, 35–28
 - Alias DEFERRED_MAPPING, 35–12
 - Alias DIRECT_MAPPING, 35–14
 - Alias HOLD_MAPPING, 35–16
 - Alias NOHOLD_MAPPING, 35–16
 - alias_deferred_mapping alias option, 35–13
 - alias_direct_mapping alias option, 35–14
 - alias_hold_mapping alias option, 35–16
 - alias_moderator_mapping alias option, 35–17, 35–18
 - alias_nohold_mapping alias option, 35–16
 - Application information
 - Syntax of, 55–8
 - AUTH_ACCESS, 49–37, 49–37
 - Example, 49–40, 49–41
 - mapping_paranoid MTA option, 39–194
 - AUTH_MAPPING alias file named parameter, 35–28
 - AUTH_REWRITE, 33–32, 33–64, 33–147, 44–3
 - mapping_paranoid MTA option, 39–194
 - Timing of application, 44–16
 - BURL_ACCESS, 49–7
 - mapping_paranoid MTA option, 39–194
 - Callout routines, 37–24
 - Callout to general database
 - Example, 37–19
 - Performance, 37–18
 - CANT_MAPPING alias file named parameter, 35–28
 - CHARSET-CONVERSION, 38–16
 - include_conversiontag MTA option, 39–190
 - Line wrapping, Compared to linelength channel option, 33–47
 - MIME relabelling, 38–25
 - Reprocess channel is invisible, 52–18
 - serviceconversion channel option as alternate trigger, 33–55
 - CHARSET-CONVERSION vs. CONVERSION timing, 38–28
 - COMMENT_STRINGS, 35–55
 - commentmap channel option, 33–66
 - sourcecommentmap channel option, 33–66
 - use_comment_strings MTA option, 39–198
 - CONVERSIONS, 38–2
 - Channel (alternate) example, 38–4, 38–5, 47–23
 - Conversion tag, 38–3
 - Example with conversion tag, 35–12
 - include_conversiontag MTA option, 39–190
 - original_channel_probe MTA option, 39–198
 - Reprocess channel is invisible, 52–18
 - DISPOSITION_LANGUAGE, 47–17
 - language channel option, 33–72, 33–96
 - Efficiency of large, 37–18
 - Entry patterns, 37–4
 - \$n substitutions, 37–6
 - Asterisk character, 37–6
 - Back match wildcards, 37–5
 - Backslash character, 37–5
 - Dollar sign character, 37–5
 - Double quote character, No special meaning, 37–5
 - Glob match example, 38–12
 - Hyphen character within glob, 37–5
 - IPv4 matching, 37–6
 - IPv6 matching, 37–7
 - Parentheses characters, No special meaning, 37–5
 - Right bracket within glob, 37–5
 - Single quote character, No special meaning, 37–5
 - Wildcard matching, Greedy or minimal, 37–6
 - Entry templates, 37–7
 - \$#sec-file-spec#, 37–11
 - \$&...&, 37–13
 - \$+n#...#, 37–12
 - \$.temporary-failure-text., 37–15

- \$=, 37-14
- \$= effect turned off by \$_, 37-9
- \$?x? random result, 37-10
- \$C metacharacter, 37-8, 37-9
- \$E metacharacter, 37-8, 37-9
- \$L metacharacter, 37-8, 37-9
- \$n substitution, 37-9
- \$R metacharacter, 37-8, 37-9
- \$[image,routine,argument], 37-17
- \$\, 37-9
- \$]ldap-url[, 37-13
- \$^, 37-9
- \$_, 37-9
- \$_ turns off LDAP URL quoting, 37-14
- \$`expression', 37-17
- \$|mapping-name;probe|, 37-16
- \$}domain-name,attribute{, 37-15
- \$}user-identifier,attribute{, 37-15
- Case of substitutions, 37-9
- Dollar sign character, 37-8
- Domain map attributes, 37-15
- Expression substitution, 37-17
- External LDAP lookup, 39-182
- extldap: URL, 39-182
- General database lookup, 37-16
- Hash substitution, 37-12
- LDAP lookup temporary failure, 37-15
- LDAP URL, 37-13
- LDAP URL encoding, 37-14
- Mapping table substitutions, 37-16
- Routine substitution, 37-17
- Sequence file, 37-11
- User identifier attributes, 37-16
- UTF-8 strings, 37-13
- ETRN_ACCESS, 33-117, 33-117
 - Restricting ETRN use, 49-51
- FILTER_testname, 5-63
- Flags
 - Testing, 37-10
- Flow of evaluation through a table, 37-9
- FORWARD, 35-59
 - \$H blocks reapplication, 35-60
 - Arbitrary LDAP attribute values in probes, 39-197
 - BSMTP channels, 50-3
 - Consulted before forward database, 35-60
 - include_conversiontag MTA option, 39-190
 - ldap_spare_N values, 39-125
- FROM_ACCESS, 44-1, 44-2
 - \$! flag, Disables sending back a vacation message, 5-45
 - \$, spam level flag, 5-42
 - \$H flag, Diagnosing .HELD files, 52-11
 - \$S flag imposing a message size limit, 33-112
 - \$S flag, error_text_recipient_over MTA option, 39-161
 - \$~ flag, 44-13, 47-23
 - \$~ for source channel switching, 33-82
 - Alternative to authentication for SMTP SUBMIT use, 33-119
 - Arbitrary LDAP attribute values in probes, 39-193
 - Disabling vacation message generation, 44-15
 - End user not seeing error text, 44-15
 - Example setting Sieve environment item, 5-28
 - include_conversiontag MTA option, 39-191
 - Initial configuration, 44-15
 - mapping_paranoia MTA option, 39-194
 - Reprocess channel, 52-19
 - Timing of application, 44-15
- from_access, 44-13
- GROUP_AUTH, 36-21
 - ldap_auth_mappingN MTA options, 39-141
 - mapping_paranoia MTA option, 39-194
- GROUP_TEMPLATES, 36-11, 36-15
 - Example, 36-12, 39-136
 - ldap_group_dn MTA option, 39-135
- HOLD_MAPPING alias file named parameter, 35-35
- INTERNAL_IP, 44-6
 - Example, 44-6
 - Example for LMTP back end, 49-15
 - Rewrite rule use of, 34-12
 - SMTP relay blocking, 49-48
- IP_ACCESS, 49-43
 - Alternative to lastresort channel option, 33-62, 33-140
 - Compared with loopcheck channel option, 33-129
 - use_ip_access MTA option, 39-194
- Large
 - Efficiency of, 37-18
- LDAP callouts
 - PORT_ACCESS, Not supported prior to JES MS 6.3-0.15, 44-5
- Location of, 40-9
- LOG_ACTION, 55-9
 - Examples, 55-11
 - Examples, Block submissions of local spambots, 55-17
 - Examples, Blocking botnet attack, 55-19
 - Examples, Disable connection transaction log entries for particular source, 55-11
 - Examples, Syslog notice upon excessive bad password SMTP AUTH attempts, 55-11

- Examples, Syslog notice upon SMTP AUTH bad username attempts, 55–13
- Examples, Syslog notice upon SMTP AUTH failures, 55–15
- Examples, Syslog notice when time-in-queue is high, 55–16
- log_deliver_by MTA option bit 1 (value 2), 39–252, 39–255
- log_futurerelease MTA option bit 1 (value 2), 39–261
- Loop in processing, 37–8
- MAC-TO-MIME-CONTENT-TYPES, 38–21
 - Sample entries, 38–22
- MAIL_ACCESS, 44–1, 44–2, 44–7
 - \$V, \$v, \$Z, \$z flags, discard or jettison actions, 5–23
 - Arbitrary LDAP attribute values in probes, 39–193
 - Deferred expansion of groups, 39–184
 - include_conversiontag MTA option, 39–191
 - mapping_paranoia MTA option, 39–194
 - Reprocess channel, 52–19
 - Reprocess channel is invisible, 52–18
 - Sieve hierarchy, 5–65
 - Timing of application, 44–15
- mapping group, 37–17
- map_names_size MTA option, 39–180, 39–197
- MESSAGE-SAVE-COPY, 54–3
 - Copy operation, 54–4
 - Example, 54–5, 54–5
 - Examples, 54–4
 - File close operation, 54–5
 - Format, 54–4
 - Job Controller notification, 54–5
 - message_save_copy_flags MTA option, 39–198
 - Rename operation, 54–4
 - Result file specification on same disk, 54–4
- MILTER_MACROS, 45–14
 - mapping_paranoia MTA option, 39–194
- MODERATOR_MAPPING alias file named parameter, 35–35
- MTA options, 39–188
 - string_pool_size_1, 39–181
- Naming convention
 - Site-supplied, 37–22
- NOHOLD_MAPPING alias file named parameter, 35–35
- NOTIFICATION_LANGUAGE
 - language channel option, 33–72, 33–96
 - preferredLanguage, 39–119
- ORIG_MAIL_ACCESS, 44–2, 44–7
 - \$V, \$v, \$Z, \$z flags, discard or jettison actions, 5–23
 - Arbitrary LDAP attribute values in probes, 39–193
 - Deferred expansion of groups, 39–184
 - ims-ms channels, 51–3
 - include_conversiontag MTA option, 39–191
 - mapping_paranoia MTA option, 39–194
 - Reprocess channel, 52–19
 - Reprocess channel is invisible, 52–18
 - Sieve hierarchy, 5–65
 - Timing of application, 44–15
- ORIG_SEND_ACCESS
 - \$V, \$v, \$Z, \$z flags, discard or jettison actions, 5–23
 - Arbitrary LDAP attribute values in probes, 39–193
 - Example, 37–3, 37–19
 - Example with SRS in use, 49–50
 - include_conversiontag MTA option, 39–190
 - mapping_paranoia MTA option, 39–194
 - Reprocess channel, 52–19
 - Reprocess channel is invisible, 52–18
 - Sieve hierarchy, 5–65
 - SMTP relay blocking, 49–48
- Performance, 37–18, 56–3
- PERSONAL_NAMES, 35–55
 - include_conversiontag MTA option, 39–191
 - personalmap channel option, 33–41, 33–77
 - sourcepersonalmap channel option, 33–41, 33–77
 - use_personal_names MTA option, 39–201
- PORT_ACCESS, 44–1, 44–2, 44–3
 - dns_verify_domain Dispatcher option interaction, 41–3
 - Example for LMTP back end, 49–15
 - mm_check_reputation callout, 45–10
 - mm_check_reputation callout, Example, 45–11
 - Timing of application, 44–15
- Pre-defined, 37–21
- Randomizing success, 37–10
- REVERSE, 35–52
 - Conversion tags and test -rewrite utility, 58–38
 - Example modifying Message-Id:, 57–3
 - include_conversiontag MTA option, 39–191
 - Limiting emission of internal host names, 57–3
- Routine callout delays
 - Logging of, 39–253
- Routine callouts
 - mm_check_reputation, 45–11

SEND_ACCESS, 44-1, 44-2, 44-2, 44-7, 44-7
 \$V, \$v, \$Z, \$z flags, discard or jettison actions, 5-23
 Arbitrary LDAP attribute values in probes, 39-193
 Deferred expansion of groups, 39-184, 39-184
 include_conversiontag MTA option, 39-190
 mapping_paranoia MTA option, 39-194
 Reprocess channel, 52-19
 Reprocess channel is invisible, 52-18
 Sieve hierarchy, 5-65
 Timing of application, 44-15, 44-15
 SEND_ACCESS, etc.
 \$, spam level flag, 5-42
 Sieve filter
 vnd.sun.source-channel environment item, 5-15
 SIEVE_EXTLISTS, 5-29
 ldap_spare_4, ldap_spare_5, ldap_spare_6 MTA option, 39-125
 mapping_paranoia MTA option, 39-194
 Site-specific
 Called from rewrite rule, 34-25
 SPF_LOCAL, 33-145
 string_pool_size_1 MTA option, 39-181
 Substitutions
 Expression, Example, 49-41
 LDAP domain specific attribute, Example, 49-41
 Mapping table, Example, 49-41
 Syntax, 37-2
 Legacy configuration, 37-2
 Unified Configuration, 37-3
 Testing of, 37-22
 test -mapping utility, 37-22
 test -match utility, 37-22, 58-26
 test -rewrite utility, 37-23, 58-28
 TLS_ACCESS, 49-45
 Transport information
 Syntax of, 55-7
 URL result mapping named by
 ldap_url_result_mapping MTA option
 Applies to ldap_group_dn MTA option, 39-136
 Applies to ldap_group_dn2 MTA option, 39-137
 USERNAME_MAPPING SpamAssassin option, 45-7
 When changes take effect, 37-21
 Wildcards
 \$n* back match, Example, 49-41
 Back match, 37-5
 Greedy or minimal, 37-6
 IPv4, 37-6
 IPv6, 37-6
 Maximum, 37-6
 test -match to test behavior of, 37-5
 wild_pool_size MTA option, 39-181
 mapping_paranoia MTA option, 39-194
 *_ACCESS mapping table probes, 44-8
 BURL_ACCESS_mapping_table, 49-8
 MILTER_MACROS mapping table, 45-14
 Mass mailings
 Performance impact, 56-2
 master channel option, 33-102
 master_debug channel option, 33-85, 49-38
 ims-ms channels, 51-6, 51-7
 os_debug MTA option, 39-75
 Reprocess channel, 33-85, 52-19
 maxage Message Store relinker option, 16-24
 maxblocks channel option, 33-48
 maxbodysize notifytarget option, 24-3
 maxcachefilesize Message Store option, 16-12
 maxcollectmsglen MSHTTP option, 29-5
 maxconcurrentconnectionattempts MMP/IMAP proxy/POP proxy option, 27-18
 maxfolders Message Store option, 16-6
 maxheaderaddrs channel option, 33-73
 maxheaderchars channel option, 33-73
 maxheadersize notifytarget option, 24-3
 maxjobs channel option, 33-99, 33-105
 ims-ms channels, 51-1
 Job Controller operation, 42-2
 Modified effect under stress, 42-4
 maxldaplimit MSHTTP option, 29-5
 maxlines channel options, 33-48
 maxlog Message Store option, 16-6
 maxlogfiles logfile option, 7-20
 maxlogfiles MMP logfile option, 7-20
 maxlogfilesize logfile option, 7-20
 maxlogs Message Store option, 16-26
 maxlogsize logfile option, 7-20
 maxmessages Message Store option, 16-7
 maxmessagesize IMAP option, 21-4
 maxmessagesize MSHTTP option, 29-10
 maxnoops IMAP option, 21-4
 maxnumberofentries PAB option, 59-2
 maxperiodicnonurgent channel option, 33-105
 maxperiodicnormal channel option, 33-105
 maxperiodicurgent channel option, 33-105
 maxpostsize MSHTTP option, 29-10
 maxprocchars channel option, 33-90
 maxprotocolerrors IMAP option, 21-12
 maxprotocolerrors POP option, 22-3
 maxsearchmailboxes IMAP option, 21-4
 maxsessions IMAP option, 21-12

maxsessions MSHTTP option, 29–12
 maxsessions POP option, 22–3
 maxthreads IMAP option, 21–12
 maxthreads Message Store purge option, 16–23
 maxthreads MeterMaid option, 46–4
 maxthreads MMP option, 27–18
 maxthreads MSHTTP option, 29–12
 maxthreads POP option, 22–3
 max_addheaders MTA option, 5–25, 39–225
 max_cache_messages Job Controller option, 42–12
 Operation under stress, 42–3
 Overriding via `imsimta cache -change -global -max_messages=N`, 58–4
 max_conns Dispatcher option, 41–6
 Operation, 41–2
 max_conns Dispatcher service option, 41–6
 max_conns MeterMaid client option, 46–4
 max_conns smpp_relay option, 53–10
 max_conns smpp_server option, 53–13
 max_conns sms_gateway option, 53–4
 max_duplicates MTA option, 39–226, 39–231
 max_entries local_table MeterMaid option, 46–3
 max_fileintos MTA option, 39–226
 max_handoffs Dispatcher option, 41–7
 max_header_blocks MTA option, 39–208
 max_header_block_use MTA option, 39–208
 max_header_lines MTA option, 39–208
 max_header_line_use MTA option, 39–208
 max_idle_time Dispatcher option, 41–7
 max_internal_blocks MTA option, 39–173
 Performance impact, 56–1
 max_life_askwork Job Controller option, 42–13
 max_life_time Dispatcher option, 41–7
 max_life_time Job Controller option, 42–13
 max_mime_levels MTA option, 39–209
 Diagnosing .HELD files, 52–11
 max_mime_parts MTA option, 39–209
 Diagnosing .HELD files, 52–11
 max_notifys MTA option, 39–226
 max_procs Dispatcher option, 41–7
 Operation, 41–2
 max_redirect MTA option
 Sieve redirect action, 5–40
 max_redirects MTA option, 39–226
 max_redirect_addresses MTA option, 39–226
 redirect to external list, 5–41
 Sieve external list example, 5–32
 max_shutdown Dispatcher option, 41–8
 max_sieve_list_size MTA option, 39–226
 max_sieve_string_size MTA option, 39–227
 max_vacations MTA option, 39–227
 vacation action, 5–44
 Vacation message not generated, 5–45
 max_variables MTA option, 39–227
 maysasl channel option, 33–149
 maysaslclient channel option, 33–149
 AUTH_ACCESS mapping, 49–38
 maysaslserver channel option, 33–149
 maytls channel option, 33–83, 33–151
 maytlsclient channel option, 33–83, 33–151
 maytlsserver channel option, 33–83, 33–151
 Should be set on SMTP SUBMIT server channel, 33–119
 mboxutil utility
 enablelastaccess base option, 7–9
 Moving (and pinning) folders to specified partition, 16–7
 Memcache, 39–228
 Errors in protocol
 Logging of, 5–63
 Message tracking and recall, 48–1
 MTA options, 39–202
 Sieve filter memcache extension, 5–53
 tracking_mode MTA option, 39–210
 Vacation messages not being generated, 5–45
 memcache: URLs
 MTA URL types, 1–4
 memcache_expire MTA option, 39–202
 Use with Message Tracking, 48–1
 memcache_host MTA option, 39–202
 check_memcache.so use of, 37–24
 Effect on duplicate_tracking_url, 39–231
 Sieve duplicate test, 5–24
 Sieve filter memcache extension, 5–53
 Use with Message Tracking, 48–1
 memcache_port MTA option, 39–202
 check_memcache.so use of, 37–24
 Effect on duplicate_tracking_url, 39–231
 Sieve duplicate test, 5–24
 Use with Message Tracking, 48–1
 memcache_timeout MTA option, 39–202
 Memory
 ims-ms channel usage
 dequeue_map, 39–172
 Job Controller usage
 max_cache_messages Job Controller option, 42–12
 LMTP server usage
 BUFFER_SIZE TCP/IP-channel-specific option, 39–173, 49–22
 MTA usage
 Internal size MTA options, 39–175
 max_internal_blocks MTA option, 39–173
 Shared
 Identifier for `ftok()` calls, projectid base option, 7–18

- Identifier for ftok() calls, projectid MTA option, 39-174
- SMTP server/SMTP SUBMIT server usage
 - max_internal_blocks MTA option, 39-173
- Message
 - .HELD
 - alias_hold_* alias options, 35-16
 - directoryscan SNMP option, 60-2
 - held_sndopr MTA option, 39-220, 39-246
 - Hold channel, Releasing, 52-11
 - loopcheck channel option, 33-128
 - Not tracked by Job Controller, 42-3
 - Archiving
 - Address reversal, 35-50
 - MESSAGE-SAVE-COPY mapping table, 54-3
 - MTA options, 39-203
 - MTA options, message_hash_fields, 39-204
 - MTA options, unique_id_template, 39-205
 - Attachments
 - CHARSET-CONVERSION mapping table, 38-20
 - Converting from non-standard formats, 38-20
 - gzipattach MSHTTP option, 29-6
 - gzippedynamic MSHTTP option, 29-6
 - gzipstatic MSHTTP option, 29-6
 - MS Mail SMTP gateway, 38-20
 - Pathworks Mail, 38-20
 - Automatic discard
 - Sieve filter discard, 5-1
 - BinHex
 - thurman channel option, 33-50
 - Body processing
 - Performance, chunk_cache_limit MTA option, 39-177
 - Performance, describe_cache_limit MTA option, 39-177
 - See also Message, Conversions, 38-1
 - Bouncing
 - return_bounced.txt, 47-12
 - Bulk precedence
 - Notification, use_precedence MTA option, 39-217
 - Capture, 54-1
 - \$M flag in *_ACCESS mapping tables, 47-2
 - Address reversal, 35-50
 - alias_capture alias option, 35-11
 - alias_journal alias option, 35-11
 - CAPTURE alias file named parameter, 35-30
 - capture Sieve action, 5-51
 - clonehosts channel option, 33-36, 33-61
 - Form of notification message, 47-2
 - Format examples, 54-7
 - Format of, 39-117
 - JOURNAL alias file named parameter, 35-30
 - Journal format, journal_format MTA option, 39-203
 - Journal format, Recipient types, 33-29, 33-108
 - LDAP attributes, 54-6
 - LDAP attributes, Address reversal, 54-6
 - ldap_capture MTA option, 39-116
 - ldap_domain_attr_capture MTA option, 39-148
 - MESSAGE-SAVE-COPY mapping table, 54-3
 - return_capture.txt, 47-12
 - Sieve external lists, 5-34
 - Sieve filters, 54-6
 - Trigger via address access mapping tables, 44-9
 - Conversions, 38-1
 - Damaged
 - BinHex blobs instead of attachments, nothurman channel option, 33-50
 - BinHex blobs instead of attachments, nouma channel option, 33-50
 - BinHex blobs instead of attachments, thurman channel option, 33-50
 - Character set interactions with forced line breaks, 33-133
 - Character set mis-labelling, 33-52
 - Forced line breaks, 33-133
 - Fragmentation, 52-3
 - Fragmentation, defragment channel option, 52-3
 - Fragmentation, Fragments timing out, 52-4
 - Fragmentation, Too many parts, 52-4
 - Header/body separation, 33-73
 - HTML, Long lines, 33-133
 - Long lines, 33-133
 - Missing = characters, 33-46
 - Truncated long line, 33-133
 - UUENCODEd blobs instead of attachments, nothurman channel option, 33-49
 - UUENCODEd blobs instead of attachments, nouma channel option, 33-50
 - UUENCODEd blobs instead of attachments, thurman channel option, 33-50
 - White space in message header lines, 33-70
 - Wrapped long line, 33-133
 - Disclaimer
 - addsuffix Sieve action, 5-49
 - Disposition
 - Language, 47-17
 - Duplicate
 - duplicate Sieve test, 5-18
 - Mailing list copy, alias_header_check alias option, 35-16

- Mailing list copy, HEADER_CHECK named parameter, 35–35
- Mailing list copy, ldap_check_header MTA option, 39–141
- Mailing list copy, Members of multiple sub-lists, 36–19
- Sieve duplicate extension, 5–24
- Sieve duplicate extension, MTA options, 39–229
- Encoding
 - CHARSET-CONVERSION mapping table, 38–17
 - CONVERSIONS mapping table, 38–3
- Envelope
 - envelopetunnel channel option, 33–68
- Envelope From
 - *-owner@*, 5–45, 44–15
 - *-request@*, 5–45, 44–15
 - Blank and the return_envelope MTA option, 39–156, 39–215
 - Disabling vacation messages back to list owner addresses, 5–45, 44–15
 - LISTSERVE@*, 5–45, 44–15
 - MAILER-DAEMON@*, 5–45, 44–15
 - majordomo@*, 5–45, 44–15
 - owner-.*@*, 5–45, 44–15
 - redirect Sieve action overriding, 5–41
 - setenvelopefrom Sieve action, 5–8, 5–18, 5–61
- Envelope To
 - NOTIFY=NEVER DSN flag, Disables sending back a vacation message, 5–45
- Expiration
 - *notices channel options, 33–96
 - alias_expiry alias option, 35–15
 - EXPIRY alias file named parameter, 35–33
 - expirysource channel option, 33–68, 33–105
- Format conversion
 - MacMIME, 38–21
 - RFC 1154, Encoding: header line, 33–46
- Forwarding
 - FORWARD mapping table, 35–59
 - ldap_forwarding_address MTA option, 39–130
 - Sieve filter redirect, 5–1
 - Sieve redirect action, 5–40
 - sieve_user_carryover MTA option, 39–101, 39–225
- Fragmentation
 - maxblocks channel option, 33–48
 - maxlines channel option, 33–48
 - max_header_block_use MTA option, 33–48, 39–208
 - max_header_line_use MTA option, 33–48, 39–208
- Fragments
 - Delivery of, 52–4
- Holding
 - alias_hold_* alias options, 35–16
 - delivery_option clause, 39–93
 - holdlimit channel option, 33–58, 33–89, 33–103, 33–113
 - HOLD_LIST alias file named parameter, 35–35
 - HOLD_MAPPING alias file named parameter, 35–35
 - mailDomainStatus of hold, 7–7, 39–145
 - mailUserStatus of hold, 39–115
 - Sieve hold action, 5–51
- Importance
 - importanceadjust and importancetest Sieve actions, 5–18
 - Sieve filter importance extension, 5–52
- List precedence
 - Notification, use_precedence MTA option, 39–217
- Manifest
 - Example constructing with Sieve, 5–39
 - Sieve filter construction of, 5–38
- MIME structure
 - Adding prefix or suffix text to first text part, addprefix and addsuffix Sieve extensions, 5–49
 - Adding prefix or suffix text to first text part, alias_prefix_text and alias_suffix_text alias options, 35–20
 - Adding prefix or suffix text to first text part, mgrpMsgPrefixText LDAP attribute, 39–140
 - Adding prefix or suffix text to first text part, mgrpMsgSuffixText LDAP attribute, 39–140
 - Boundary markers, 33–45
 - Converting from non-standard formats, 38–20
 - Damaged, Illegal encoding, 33–46
 - message/partial, defragment channel option, 33–46
 - message/partial, Defragmentation channel, 52–3
 - multipart/encrypted, 33–45
 - multipart/signed, 33–45
 - Redundant multipart levels, CHARSET-CONVERSION mapping table, 38–17
 - Redundant multipart levels, CONVERSIONS mapping table, 38–3
- Monitoring
 - MESSAGE-SAVE-COPY mapping table, 54–3
- Notification, 47–1

Alias expansion value used in, 35–24

Capture, 47–2

Capture, \$M flag in *_ACCESS mapping tables, 47–2

Channel options, 33–93

Delay warning, DSN SMTP extension, 33–96, 33–131

Delayed, use_precedence MTA option, 39–217

Delivery delay warning, return_delayed.txt, 47–12

Delivery delay warning, See also notices channel option, 47–12

Delivery receipt, 47–1

Delivery receipt, return_delivered.txt, 47–13

Delivery receipt, return_forwarded.txt, 47–13

Disposition, 47–1

Enqueued to process channel, 52–18

Envelope From address, 47–24

filter_discard channel messages not eligible, 52–8

Format of, 47–4

Format of, DSN language, 47–8

Format of, history_to_return MTA option, 39–213

Format of, LOG_BANNER TCP/IP-channel-specific option, 49–27

Format of, LOG_TRANSPORTINFO TCP/IP-channel-specific option, 49–28

Format of, MDN language, 47–17

FROM_ACCESS mapping rejection text, 44–15

Generated by remote MTAs, 47–3

Generated by SMTP clients, 47–2

Generation of, 47–3

Generation of, Scheduler, 8–4

Generation, Channel used, 33–95

Group (in LDAP) syntax errors, 47–2

Language choice, DISPOSITION_LANGUAGE mapping table, 47–17

Language choice, language channel option, 33–72, 33–96

Language choice, NOTIFICATION_LANGUAGE mapping table, 47–8

Language preference, Address reversal, 35–50

Language specific, RETURN_PERSONAL option in return_option.opt, 39–216

ldap_delay_notifications MTA option, 39–138

Localization, langdir MTA option, 39–154

Logging of, 47–24

log_message_id MTA option, 47–24

log_process MTA option, 47–24

Mailing lists, 36–17

Mailing lists, alias_envelope_from alias option, 35–14

Mailing lists, alias_error_text alias option, 35–14

Mailing lists, alias_keep_delivery alias option, 35–17

Mailing lists, alias_keep_read alias option, 35–17

MDN, disposition_deleted.txt file, 47–20

MDN, disposition_dispatched.txt file, 47–20

MDN, disposition_prefix.txt file, 47–20

MDN, disposition_suffix.txt file, 47–20

msprobe alarm messages, 11–1

MTA options, 39–213

Non-delivery report, 47–1

Nondelivery, return_bounced.txt, 47–12

Nondelivery, return_failed.txt file, 47–13

Nondelivery, return_timedout.txt, 47–13

nonotify Sieve action, 5–18

NOTARY flags, Address reversal, 35–50

notary_decode MTA option, 39–214

Over quota warnings, 47–2

Overquota in Message Store, 47–3

Postmaster, 47–25

Postmaster address, Domain-specific, 39–148

Postmaster address, returnaddress channel option, 33–97

Postmaster address, returnpersonal channel option, 33–97

Postmaster address, return_personal MTA option, 39–216

Postmaster copied on, 47–1

Postmaster notifications of channel and system Sieve filter syntax errors, 47–2

Postmaster, Format of msprobe alarm messages, 11–2

Process channel, 47–24

Read receipt, 47–1

Request, alias_[no]delay_notifications alias options, 35–13

Return job, return_units MTA option, 39–216

Return of content, 47–25

Return of content, content_return_block_limit MTA option, 39–207, 39–213

Return of content, DSN SMTP extension, 33–96, 33–131

Return-of-content flag, ldap_blocklimit MTA option, 39–123

return_envelope MTA option, 39–155, 39–215

return_option.opt file, 47–13

return_prefix.txt, 47–12

Routing of, 47–22

- setnotify Sieve action, 5–18
- setreturn Sieve action, 5–18
- Sieve filter control of DSN settings, 5–61
- Sieve filter notify, 5–1
- Sieve filter processing error, return_error.txt, 47–13
- Sieve filter syntax error reports, 47–2
- Sieve filters, redirect-dsn extension, 5–41
- Sieve reject extension, 5–28
- Sieve syntax error, 39–226, 39–227
- Size limit, blocklimit channel option, 39–207, 39–213
- Size limit, content_return_block_limit MTA option, 39–207, 39–213
- Size limits, 39–206, 39–213, 47–25
- Spam bounces, 47–23
- SRS addresses, Relay blocking interaction, 49–50
- Subject: header line, 47–10
- Troubleshooting "incomplete" DSNs, 47–9
- Types, 47–1
- Vacation, 47–1
- Warning of delayed delivery, 47–1
- Priority
 - alias_priority alias option, 35–17
 - Defragmentation, 52–4
 - Effect on delivery retry frequency, 33–101
 - Effect on MTA processing, 42–2, 42–4
 - Effect on timing of DSN generation, 33–97
 - Effect on timing of message return (bounce), 33–97
 - Example of "off-hours" delivery eligibility, 42–17
 - LOG_ACTION mapping table probes, 55–10
 - Mass mailings, 36–9
 - Message size influence, 33–114, 39–209, 39–219
 - Message transaction log entries, log_priority MTA option, 39–265
 - Overriding, 42–6
 - Rewrite rule access to, 34–33
 - SMS messages, use_sms_priority SMS gateway option, 53–8
 - [PRIORITY] named parameter for mailing lists, 35–35
- Priority (MT-PRIORITY)
 - CONVERSIONS mapping table probe, include_mtpriority MTA option, 38–3
 - envelopetunnel channel option, 33–68
 - LOG_ACTION mapping table probes, log_mtpriority MTA option, 39–264
 - Mapping table probes, 39–191
 - Message transaction log entries, log_mtpriority MTA option, 39–264, 39–266
 - MESSAGE-SAVE-COPY mapping table probes, message_save_copy_flags MTA option, 39–198
 - Policy, mtpriority_policy MTA option, 39–219
 - vnd.oracle.mt-priority Sieve environment item, 5–27
- Queue files
 - Creation of, addrserverfile channel option, 33–57
 - Creation of, expandchannel channel option, 33–58, 33–89, 33–103, 33–113
 - Creation of, expandlimit channel option, 33–58, 33–89, 33–103, 33–113
 - Creation of, multiple channel option, 33–57
 - Creation of, osync MTA option, 39–174
 - Creation of, single channel option, 33–57
 - Creation of, single_sys channel option, 33–57
 - Creation of, subdirs channel option, 33–59
 - Flushing of, fsync MTA option, 39–172
- Recall of
 - See Message recall, 48–1
- Recipients
 - Access control mapping tables, 44–7
 - Limiting number of, Address access mapping tables, 44–9
 - Limiting number of, ALLOW_RECIPIENTS_PER_TRANSACTION, 49–18
 - Limiting number of, Channel options, 33–87, 33–120
 - Limiting number of, ldap_domain_attr_recipientcutoff MTA option, 39–150
 - Limiting number of, ldap_domain_attr_recipientlimit MTA option, 39–150
 - Limiting number of, ldap_recipientcutoff MTA option, 39–117
 - Limiting number of, ldap_recipientlimit MTA option, 39–117
- Reformatting, 38–1
- Replay
 - MESSAGE-SAVE-COPY mapping table, 54–5
- Returning
 - return_bounced.txt, 47–12
- Routing
 - See Routing, 38–4
- Sieve notify
 - notify_maximum_timeout MTA option, 39–65
 - notify_minimum_timeout MTA option, 39–66
- Size

- Affected by encoding, decoding, and conversions, 39–192
- Logging of, Reported in units of MTA
 - block_size, 39–206
 - Logging of, sz attribute, 39–258
- Mapping table probes, include_mtpriority MTA option, 39–191
- Rewrite rule access to, 34–33
- Sieve filter access to, 5–12, 5–38
- SMTP SIZE extension, Interaction with size limit settings, 39–159
- SMTP SIZE extension, Mapping table probes, 39–192
- Size limits, 33–112
 - Address reversal, 35–50
 - alias_blocklimit alias option, 35–11
 - alias_linelimit alias option, 35–11
 - BLOCKLIMIT alias file named parameter, 35–30
 - Bounce messages, 39–206, 39–213
 - Bounce messages, content_return_block_limit MTA option, 39–207, 39–213
 - Force non-return of content flag, 47–25
 - Headers, header_limit MTA option, 39–207
 - Headers, max_header_blocks MTA option, 39–208
 - Headers, max_header_lines MTA option, 39–208
 - ldap_blocklimit MTA options, 39–123
 - ldap_domain_attr_blocklimit MTA option, 39–145
 - ldap_domain_attr_sourceblocklimit MTA option, 39–149
 - ldap_maximum_message_size MTA option, 39–133
 - ldap_message_quota MTA option, 39–125
 - ldap_sourceblocklimit MTA option, 39–118
 - LINELIMIT alias file named parameter, 35–30
 - Logging rejection due to, 33–113
 - mailDomainMsgMaxBlocks domain LDAP attribute, 39–145
 - mailMsgQuota attribute, 39–125
 - maxmessagesize IMAP option, 21–4
 - maxmessagesize MSHTTP option, 29–10
 - maxpostsize MSHTTP option, 29–10
 - max_header_block_use MTA option, 33–48
 - max_header_line_use MTA option, 33–48
 - MTA options, 39–205
 - Set via address access mapping tables, 44–9
- Text parts
 - Character set, 38–16
 - Character set labelling, 33–52
- Tracking of
 - See Message tracking, 48–1
 - Tunnelling, 50–1
- uuencode
 - thurman channel option, 33–50
 - uma channel option, 33–50
- Vacation
 - Addresses recognized,
 - ldap_autoreply_addresses MTA option, 39–128
 - Check start and end time via delivery_option, 39–93
 - delivery_option clause to use Sieve vacation action, 39–93
 - Format of, 47–8
 - Format of, ldap_autoreply_mode MTA option, 39–126
 - FROM_ACCESS mapping table disables generation of, 44–15
 - Language choice, language channel option, 33–72, 33–96
 - novacation Sieve action, 5–18
 - Repeat of, autoreply_timeout_default MTA option, 39–65
 - Repeat of, ldap_autoreply_timeout MTA option, 39–128
 - Repeat of,
 - ldap_domain_attr_autoreply_timeout MTA option, 39–147
 - Repeat of, vacation_template MTA option, 39–67
 - Sieve filter vacation, 5–1
 - Sieve filter vacation extension, 5–43
 - Subject of, ldap_autoreply_subject MTA option, 39–126
 - Text of, ldap_autoreply_text MTA option, 39–127
 - Text of, ldap_autoreply_text_internal MTA option, 39–128
 - Text of, vnd.sun.autoreply-internal envelope item in Sieve filters, 5–27
 - Time range, ldap_end_date MTA option, 39–123
 - Time range, ldap_start_date MTA option, 39–122
 - Time range, vacationEndDate LDAP attribute, 39–123
 - Time range, vacationStartDate LDAP attribute, 39–122
 - vacation_maximum_timeout MTA option, 39–66, 39–102
 - vacation_minimum_timeout MTA option, 39–66, 39–102
 - Why not generated, 5–45

- Why not generated, Domain not properly defined in LDAP or not found, 5–45
- Why not generated, Domain, user or group status, 5–45
- Why not generated, Incompatible other Sieve action performed, 5–46
- Why not generated, mailAutoreplyText LDAP attribute or value missing, 5–46
- Why not generated, Original message a list post per header lines, 5–45
- Why not generated, Original message disabled all notifications, 5–45
- Why not generated, Original message's From address suggests list post, 5–45
- Why not generated, Outside vacationStartDate-vacationEndDate range, 5–45
- Why not generated, Recipient address not present in original message header, 5–45
- Why not generated, Recipient not properly defined in LDAP or not found, 5–45
- Why not generated, Same vacation response already sent recently, 5–45
- Why not generated, Sieve novacation or FROM_ACCESS \$! applied, 5–45
- Why not generated, Sieve vacation action syntax error, 5–46
- Why not generated, Too many vacation actions already performed in Sieve script, 5–45
- Why not generated, Trouble accessing vacation-previous-response database, 5–45
- Why not generated, Vacation action not supported in system Sieves, 5–46
- Message circuit check
 - Display via XCIR SMTP command
 - DISABLE_CIRCUIT TCP/IP-channel-specific option, 49–24
- Message conversions
 - Character set conversion
 - Example, 38–19
 - Character set, Conversion, 38–18
 - Performance impact, 56–2
 - Reformatting, 38–20
- Message recall, 48–1
 - Configuration, 48–1
 - nottracking* channel options, 33–91
 - tracking* channel options, 33–91
- Message replay
 - MESSAGE-SAVE-COPY mapping table, 54–3
- Message return job
 - See return_job, 8–3, 8–4
- Message Store, 1
- Admin user
 - admins Message Store option, 16–9
 - httpproxyadmin MSHTTP option, DEPRECATED, 29–7
 - indexeradmins Message Store option, 16–9
 - proxyadmin base option, 7–15
 - proxyadminpass base option, 7–15
 - smtpauthpassword MSHTTP option, 29–6
 - smtpauthuser MSHTTP option, 29–6
 - storeadmin MMP/IMAP Proxy/POP Proxy/vdomain option, 27–27
 - storeadminpass MMP/IMAP Proxy/POP Proxy/vdomain option, 27–27
- Backup
 - See also backup_group options, 19–1
- Database snapshot job
 - Scheduler task, 8–2
- Database snapshot verification job
 - Scheduler task, 8–2
- Database transactions
 - dbtxnsync base option, 7–9
 - maxlog Message Store option, 16–6
- Delivery channel
 - See ims-ms channels, 51–1
- Disk space
 - relinker, 16–24
- Expiration job
 - Scheduler task, 8–2
- File creation
 - umask Message Store option, 16–13
- Folders
 - Delivery to, 33–42, 33–111
 - Event notification, noninbox notifytarget option, 24–5
 - Hidden, ensureownerrights Message Store option, 16–6, 16–6
 - Maximum age of messages (days), messagedays store.expirerule option, 16–19
 - Maximum retention (days) for over-sized messages, messagesizedays store.expirerule option, 16–19
 - Maximum size of (# messages), messagecount store.expirerule option, 16–19
 - Maximum size of (bytes), foldersizebytes store.expirerule option, 16–19
 - Maximum size of message (bytes), messagesize store.expirerule option, 16–19
 - Shared, proxyserverlist base option, 7–16
- Hierarchical storage
 - Pinning folders onto specified partitions, 16–7
- Logging
 - logexpungedetails Message Store option, 16–6
 - maxlog Message Store option, 16–6

-
- Mailbox locked
 - MAILBOX_BUSY_FAST_RETRY TCP/IP-channel-specific option, 49–29
 - Message expiration
 - Age at which purge permanently removes, cleanupage Message Store option, 16–10
 - Sieve filters, 5–1
 - Message typing
 - contenttype Message Store message type option, 16–21
 - enable Message Store message type option, 16–21
 - enable Message Store typequota option, 16–21
 - flagname Message Store message type option, 16–22
 - header Message Store message type option, 16–21
 - msgtypes notifytarget option, 24–5
 - Options, 16–20
 - quotaroot Message Store message type option, 16–22
 - Quotas, 16–21
 - Options
 - See Message Store options, 16–3
 - Partitions
 - defaultpartition Message Store option, 16–11
 - diskusage threshold Message Store option, 16–5
 - Pinning folders, 16–7
 - See also Partition options, 18–1
 - Performance
 - dbtmpdir Message Store option, 16–11
 - relinker, 16–24
 - Shared folders, 16–25
 - Snapshot
 - snapshotdirs Message Store option, 16–8
 - snapshotpath Message Store option, 16–9
 - Transaction log
 - finalcheckpoint option, 16–6
 - Utilities
 - mboxutil, Moving (and pinning) folders to specified partition, 16–7
 - Welcome message for new users
 - Localized, welcomemsg message_language option, 17–1
 - welcomemsg base option, 7–17
 - Message Store options, 16–3
 - admins, 16–9
 - archive, 16–13
 - compliance, 16–14
 - destination, 16–14
 - intext, 16–15
 - operational, 16–14
 - path, 16–15
 - postdatedmode, 16–15
 - reportdir, 16–15
 - retrieveport, 16–15
 - retrieveserver, 16–15
 - retrievetimeout, 16–15
 - source_channel, 16–14
 - style, 16–14
 - tmpdir, 16–14
 - useheaderrecipients, 16–15
 - autorepair, 16–9
 - autorepairdebug, 16–10
 - backupdir, 16–4
 - backupexclude, 16–9
 - cachepreviewlen, 16–12
 - cachesynclevel, 16–5
 - checkdiskusage, 16–5
 - checkmailhost, 16–5
 - checkpoint, 16–16
 - debug, 7–22, 16–16
 - stresslimit, 16–16
 - cleanupage, 16–10
 - cleanupsize, 16–10
 - dbcachesize, 16–10
 - dblogregionmax, 16–10
 - dbnumcaches, 16–5
 - dbregionmax, 16–10
 - dbreplicate, 16–16
 - ackpolicy, 16–17
 - acktimeout, 16–17
 - dbpriority, 16–16
 - dbremotehost, 16–16
 - enable, 16–16
 - port, 16–16
 - queuemax, 16–17
 - twosites, 16–17
 - dbsync, 16–5
 - dbtmpdir, 16–11
 - deadlock, 16–17
 - autodetect, 16–17
 - checkinterval, 16–17
 - deadlockaggressive, 16–5
 - defaultmailboxquota, 16–11
 - defaultmessagequota, 16–11
 - defaultpartition, 16–11
 - Deleted, 16–26
 - defaultacl, 16–26
 - diskflushinterval, 16–26
 - expirestart, 16–26
 - listrecover, 16–26
 - mailboxexpungesize, 16–26
 - maxlogs, 16–26
 - message type plugin, 16–26

- serversidewastebasket, 16–26
 - snapshotdirs, 16–26
- diskflushinterval
 - Analogous to fsync MTA option, 39–172
- diskusagethreshold, 16–5
 - checkdiskusage interaction, 16–5
- enable, 16–4
 - Default for schedule.task:expire.enable, 8–3, 16–18
 - Default for schedule.task:snapshot.enable, 8–4, 8–5
- encryptnew, 16–12
- ensureownerrights, 16–6
- expire, 16–18, 16–18
 - Deleted, cleanonly, 16–27
 - Deleted, workday, 16–27
 - exploglevel, 16–18
- expirerule
 - deleted, 16–19
 - exclusive, 16–19
 - folderpattern, 16–19
 - foldersizebytes, 16–19
 - messagecount, 16–19
 - messagedays, 16–19
 - messagesize, 16–19
 - messagesizedays, 16–19
 - seen, 16–20
- expiresieve, 16–13
- expungesynclevel, 16–6
- finalcheckpoint, 16–6
- folderlockcount, 16–11
- folderquota, 16–20
 - enable, 16–20
- indexeradmins, 16–9
- indexsynclevel, 16–6
- keylabel, 16–12
- keypass, 16–12
- listimplicit, 16–6
- logexpungedetails, 16–6
- mailboxexpungesize
 - Deleted; see cleanupsize instead, 16–27
- mailboxpurgedelay, 16–12
- maxcachefilesize, 16–11
- maxfolders, 16–6
- maxlog, 16–6
- maxmessages, 16–7
- messagesynclevel, 16–7
 - Analogous to fsync MTA option, 39–172
- messagetype, 16–20
 - enable, 16–21
 - header, 16–21
 - mtindex, contenttype, 16–21
 - mtindex, flagname, 16–22
 - mtindex, quotaroot, 16–22
 - mtindex, quotaroot example, 16–21
- msghash, 16–22
 - dbcachesize, 16–22
 - enable, 16–22
 - nummsgs, 16–22
- overquotastatus, 16–7
 - Enables quota overdraft, 16–8
 - Implies quotaoverdraft, 16–8
- perusersynclevel, 16–7
- pin, 16–7
- privatesharedfolders
 - restrictanyone, 16–25
 - restrictdomain, 16–25
 - shareflags, 16–25
- publicsharedfolders
 - user, 16–25
- purge, 16–22
 - count, 16–23
 - enable, 16–23
 - maxthreads, 16–23
 - percentage, 16–23
- quotaenforcement, 16–12
- quotaexceededmsg, 16–13
- quotaexceededmsginterval, 16–12
 - Notification that a Message Store user is overquota, 47–3
- quotagraceperiod, 16–12
- quotanotification, 16–12
- quotaoverdraft, 16–8
 - Notification that a Message Store user is overquota, 47–3
- quotawarn, 16–13
 - Notification that a Message Store user is overquota, 47–3
- relinker, 16–24
 - enable, 16–24
 - maxage, 16–24
 - minsize, 16–24
- rollingdbbackup, 16–8
- seenckpinterval, 16–8
- seenckpstart, 16–8
- serviceadmingroupdn, 16–13
- Shared folders, 16–25
- sharedfolders, 16–8
- snapshotdirs, 16–8
- snapshotpath, 16–8
- subscribesynclevel, 16–9
- synclevel, 16–9
- typequota
 - enable, 16–21, 16–21, 16–22
- umask, 16–13

Message tracing, 23–1, 33–86

- messagecount Message Store expirerule option, 16–19
 - messagedays Message Store expirerule option, 16–19
 - messagesize Message Store expirerule option, 16–19
 - messagesizedays Message Store expirerule option, 16–19
 - messagesynclevel Message Store option, 16–7
 - Analogous to fsync MTA option, 39–172
 - messagetrace options, 23–1
 - actionattributes, 21–11, 23–1
 - actions, 21–11, 23–1
 - activate, 23–1, 33–86
 - ims-ms channel debugging, 51–6, 51–7
 - loglevel, 23–1, 39–72
 - messagetype Message Store options
 - mtindex
 - filename, 16–22
 - quotaroot, 16–22
 - messagetypeplugin Message Store deleted option, 16–26
 - message_hash_fields MTA option
 - Message identifier generation (for archiving), 54–19
 - message_language options, 17–1
 - quotaexceededmsg, 17–1
 - welcomemsg, 17–1
 - message_save_copy_flags MTA option, 39–198
 - MESSAGE-SAVE-COPY mapping table probe, 54–4
- Messenger Express
 - Address search
 - allowldapaddresssearch MSHTTP option, 29–4
- Meta-group
 - Mailing list definitions, 39–100
- MeterMaid, 44–1, 46–1
 - enable_sieve_metermaid MTA option, 39–229
 - IMAP pwexpirealert options
 - metermaidtable, 21–13
 - viametermaid, 21–13
 - metermaidtable pwexpirealert option, 21–13
 - MTA options, 39–211
 - Sieve filter metermaid extension, 5–57
 - Startup, 46–2
 - viametermaid pwexpirealert option, 21–13
- MeterMaid client
 - Options, 46–4
 - connecttimeout, 46–4
 - debug, 7–22, 46–2
 - max_conns, 46–4
 - remote_table, 46–3
 - remote_table, server_nickname, 46–5
 - server_host, 46–4
 - server_port, 46–5
 - timeout, 46–5
- MeterMaid options, 46–1
 - async, 46–4
 - backlog, 46–4
 - enable, 46–2
 - listenaddr, 46–5
 - local_table
 - block_time, 46–2
 - data_type, 46–2
 - inactivity_time, 46–3
 - max_entries, 46–3
 - quota, 46–3
 - quota_time, 46–3
 - resubmit_time, 46–2
 - storage, 46–3
 - table_options, 46–3
 - table_type, 46–3
 - value_type, 46–3
 - maxthreads, 46–4
 - secret, 46–5
- MeterMaid remote_server options
 - server_port, 46–5
- MeterMaid server
 - msprobe probe of, 10–2
- metermaid: URLs
 - MTA URL types, 1–4
- metermaidtable pwexpirealert option, 21–13
- metermaid_backoff MTA option, 39–211
- metermaid_expire MTA option, 39–211
- metermaid_host MTA option, 39–211
- metermaid_port MTA option, 39–211
- metermaid_secret MTA option, 39–211
- metermaid_timeout MTA option, 39–212
- MIB variables, 55–24
 - Channel counters, 55–20
 - mtaGroupLoopsDetected
 - directoryscan SNMP option, 60–2
 - mtaGroupOldestMessageId
 - directoryscan SNMP option, 60–2
 - mtaGroupOldestMessageStored
 - directoryscan SNMP option, 60–2
- Microsoft® Exchange
 - Journal format

- ;format-journal* tags on user LDAP capture attribute, 39–117
 - addrtypescan* channel options, 33–29, 33–108
 - capture_format_default MTA option, 39–92
 - DESTINATION Archive option, 45–9
 - destination Message Store archive option, 16–14
 - journal_format MTA option, 39–96, 39–204
 - SOURCE_CHANNEL Archive option, 45–9
 - source_channel Message Store archive option, 16–14
 - STYLE Archive option, 45–9
 - style Message Store archive option, 16–14
 - msexchange channel option, 33–49, 33–130, 33–152
- migrate415 PAB option, 59–2
- Migration of users
 - Hold channel, 52–10
- Milter
 - See Spam/virus filter package integration, Milter, 45–5, 45–11, 45–14
- MIME labelling
 - Batch SMTP, 50–3
 - Boundary parameter segmentation, 33–51, 33–54
 - Calendar parts
 - msexchange channel option, 33–49, 33–130, 33–152
 - charset
 - RFC 2231 encoding removal, 33–51, 33–54
 - Charset, 33–52
 - Encoding
 - Long lines, 33–47
 - Text parts, CHARSET-CONVERSION mapping table, 38–18
 - Illegally encoded message parts, 33–46
 - Illegally encoded multiparts, 33–46
 - MacMIME conversions, 38–21
 - Modifying (relabelling), 38–25
 - RELABEL conversion entry parameter, 38–8
 - Parameter encoding
 - parameterformat* channel options, 33–50, 33–54
 - rfc2231compliant MSHTTP option, 29–5
 - Parameter length limits, 33–49
 - RFC 2231 segmentation of long values, 33–50, 33–54
- minperiodicnonurgent channel option, 33–105
- minperiodicnormal channel option, 33–105
- minperiodicurgent channel option, 33–105
- minsize Message Store relinker option, 16–24
- min_conns Dispatcher option, 41–6
 - Operation, 41–2
- min_procs Dispatcher option, 41–8
 - Operation, 41–2
- Miscellaneous mapping table MTA options, 39–196
- missingrecipientpolicy channel option, 33–39, 33–74
- missing_address MTA option, 39–272
- missing_recipient_group_text MTA option, 39–58
- missing_recipient_policy MTA option, 39–58
- MLS (Multi Layer Security)
 - Channel options, 33–93
 - mllabel channel option, 33–93
 - mllrange channel option, 33–93
 - MTA options, 39–212
- mlls MTA option, 39–212
- mllabel channel option, 33–93
- mllrange channel option, 33–93
- MMP, 1
 - Access filters, 44–1
 - Denial-of-service
 - ldappendingoplimit, 27–16
 - stressfdwait base option, 7–5
 - stressperiod base option, 7–5
- Logging
 - maxlogfiles option, 7–20
- Options, 27–3
 - authcachettl, 27–5
 - banner, 27–6
 - canonicalvirtualdomaindelim, 27–9
 - connecttimeout, 27–10
 - connlimits, 27–10
 - connrejectthreshold, 27–11
 - crams, 27–11
 - debugkeys, 27–11
 - defaultdomain, 27–12
 - domainsearchformat, 27–14
 - enable, 27–5
 - hosteddomains, 27–15
 - ipv6in, 7–17, 27–15
 - ipv6out, 7–18, 27–16
 - ipv6sortorder, 7–18
 - langlist, 27–16
 - ldapcachesize, 27–16
 - ldapcachettl, 27–16
 - ldappendingoplimit, 27–16
 - ldaprefreshinterval, 27–16
 - ldaptimeout, DEPRECATED, 27–17
 - ldapurl, 27–17
 - ldapurl, DEPRECATED, 27–17
 - logfile, rollovertime, 7–20
 - loglevel, 27–17, 39–72
 - mailhostattrs, 27–18
 - maxconcurrentconnectionattempts, 27–18
 - maxthreads, 27–18
 - numprocesses, 27–18

- numthreads, DELETED; see maxthreads, 27–19
- polldelay, 21–4, 27–19
- preauth, 27–20
- preferpoll, 7–15, 27–20
- replayformat, 27–20
- replaypass, 27–21
- restrictplainpasswords, 27–21
- searchformat, 27–21
- smtpproxypassword, 27–22, 39–218
- ssladjustciphersuites, 7–11, 27–24
- sslcachedir, 7–3, 27–25
- sslsecmodfile, DELETED, 27–27
- storeadmin, 27–27
- storeadminpass, 27–27
- tcpaccess, 27–28
- tcpaccessattr, 27–28
- tcpaccessattr, TCP wrappers, 6–2
- timeout, 27–28
- usergroupdn, DEPRECATED; see ugldapbasedn instead, 27–29
- use_nslog, 27–28
- virtualdomaindelim, 27–29
- virtualdomainfile, DELETED; see vdomain options instead, 27–29
- Ports
 - tcp_listen block, 27–5
- SMTP proxy
 - PROXY_PASSWORD TCP/IP-channel-specific option, 49–31
- Startup, 27–5
- XPEHLO reset of connection
 - PORT_ACCESS mapping probe, 44–16
- MMP options
 - storeadmin
 - Default taken from store.admins option, 16–9
- mm_debug MTA option, 39–74
 - \$U flag in address *_ACCESS mapping tables, 44–9
 - \$U flag in PORT_ACCESS mapping table, 44–4
- mm_mbc MTA option, 39–173
- mm_mbf MTA option, 39–173
- Monitoring
 - MTA, 55–1
- msadmin
 - Debugging
 - debug msadmin option, 7–21
 - debugkeys values, 27–11
 - Options
 - debug, 7–21
- msconfig utility
 - Commands
 - EDIT CONVERSIONS, 38–6
 - EDIT CONVERSIONS, 39–69
 - EDIT MAPPINGS, 37–3
 - EDIT REWRITES, 34–2
 - Used to configure Messaging Server, 1
 - msexchange channel option, 33–49, 33–130, 33–152
 - msgflags notifytarget option, 24–3
 - msgtypes notifytarget option, 24–5
 - MSHTTP, 1
 - Autorestart
 - autorestart.enable option, 7–23
 - DNS reverse lookup
 - dnsresolveclient base option, 7–16
 - msprobe probe, 10–2
 - SSL
 - enableslport MSHTTP option, 29–3
 - sslport MSHTTP option, 29–13
 - sslport MSHTTP sieve option, 29–23
 - Startup, 29–3
 - MSHTTP options, 29–3
 - allowanonymouslogin, 29–11
 - allowcollect, 29–5
 - allowldapaddresssearch, 29–4
 - altservice, 29–11
 - cert_enable, 29–7
 - cert_port, 29–7
 - charsetvalidation, 29–4
 - connlimits, 27–10
 - cookiedomain, 29–8
 - cookieName, 29–4
 - da_host, 29–7
 - da_port, 29–8
 - detectcharset, 29–4
 - domainallowed, 27–13, 29–12
 - domainnotallowed, 27–14, 29–12
 - enable, 29–3
 - enableslport, 29–3
 - enableuserlist, 29–12
 - extrauserldapattrs, 29–8
 - feedback, 29–13
 - notspam, 29–14
 - spam, 29–14
 - filterhiddenmailinglists, 29–4
 - forcenbsptospace, 29–4
 - forcetelemetry, 29–12
 - fullfromheader, 29–8
 - generatereceivedheader, 29–5
 - gzipattach, 29–6
 - gzipdynamic, 29–6
 - gzipstatic, 29–6
 - htmlprocessor, 29–13
 - ICAP service use enabled, 32–1
 - httpcharset and mailcharset options, 29–14
 - httpproxyadmin, 29–7

- httpproxyadminpass, 29-7, 29-7
- idletimeout, 29-12
- ims5compat, 29-5
- ipsecurity, 29-8
- ldapaddresssearchattrs, 29-5
- logunauthsession, 29-13
- maxcollectmsglen, 29-5
- maxldaplimit, 29-5
- maxmessagesize, 29-10
- maxpostsize, 29-10
- maxsessions, 29-12
- maxthreads, 29-12
- nofilecache, 29-5
- numprocesses, 29-12
- plaintextconvspace, 29-7
- plaintextmncipher, 27-19, 29-12
- plaintexttabsize, 29-7
- popbindaddr, 29-5
- port, 29-4
- replayformat, 27-21, 29-4
- resourcetimeout, 29-10, 29-10
- rfc2231compliant, 29-5
- showunreadcounts, 29-6
- sieve, 29-23
 - port, 29-23
 - sslport, 29-23
- singlesignoff, 29-8
- smtpauthpassword, 29-6
- smtpauthuser, 29-6
- smtpghost, 29-10
- smtpport, 29-11
- smtppls, 29-11
- sourceurl, 29-11
- spooldir, 29-11
- sslcachesize, 29-13
- sslnicknames, 27-26, 29-13
- sslport, 29-13
- sslsourceurl, 29-11
- sslusessl, 29-13
- sso_enable, 29-8
- sso_id, 29-9
- sso_prefix, 29-9
- usesentdate, 29-6
- uwcontexturi, 29-9
- uwcenabled, 29-9
- uwchome, 29-9
- uwclogouturl, 29-10
- uwcport, 29-9
- uwcsslport, 29-9
- xmailer, 29-6

msprobe, 1

- crontab Scheduler task option, 8-3, 10-2
- Enable scheduling of, 10-1

Options, 10-1

- queuedir, 10-1
- timeout, 10-1
- warningthreshold, 10-2

Restart of stored

- maxlog Message Store option, 16-6

Scheduler task, 8-2

MTA, 1

- Startup, 39-7, 39-54

MTA counters, 55-20

Binning of

- log_delay_bins MTA option, 39-70
- log_size_bins MTA option, 39-70

Channel counters, 55-20

Implementation of, 55-24

- log_delay_bins MTA option, 39-70
- log_frustration_limit MTA option, 39-70
- log_size_bins MTA option, 39-70
- log_statistics MTA option, 39-70

MTA options

- log_delay_bins, 39-70
- log_frustration_limit, 39-70
- log_size_bins, 39-70
- log_statistics, 39-70

Purpose and use of, 55-24

Sieve filters, 5-50

SNMP subagents to serve out, 55-24

Strict creation of

- log_statistics MTA option, 39-70

Updating of

- log_frustration_limit MTA option, 39-70

MTA message transaction log file

- See Logging, MTA transaction, 55-1

MTA option file

- Comment characters, 39-9
- comment_chars MTA option does not affect, 39-172

MTA options, 39-7

- .HELD messages, 39-220, 39-220, 39-246

Access mapping tables, 39-188

- access_auth, 39-188
- access_errors, 39-156
- access_orcpt, 39-189
- include_connectioninfo, 39-189
- include_conversiontag, 39-190
- include_mtpriority, 39-191
- include_spares, 39-192
- include_spares1, 39-193
- mapping_paranoia, 39-194
- use_ip_access, 39-194

access_auth, 39-188

access_counts, 39-188

- *_ACCESS mapping table probes, 44-7

access_errors, 39–156
 Recipient *_ACCESS mapping table
 rejections, 44–8
 Spam/virus filter package recipient rejections,
 39–157

access_orcpt, 39–189
 *_ACCESS mapping table probes, 44–7
 SRS and relay blocking, 49–50

Alias and address, 39–54
 alias_case, 39–55
 alias_domains, 39–56
 alias_magic, 39–56
 ap_debug, 39–73
 delimiter_char, 39–57
 exproute_forward, 39–57
 improute_forward, 39–57
 max_alias_levels, 39–58
 missing_recipient_group_text, 39–58
 missing_recipient_policy, 39–58
 name_table_name, 39–59
 reverse_envelope, 39–60
 subaddress_char, 39–60
 user_case, 39–64
 use_alias_database, 39–60
 use_auth_return, 39–194
 use_canonical_return, 39–194
 use_domain_database, 39–61
 use_forward_database, 39–61, 39–199
 use_orig_return, 39–194
 use_reverse_database, 39–62, 39–200

aliasdetourhost_null_optin, 39–91

Aliases in LDAP, 35–5
 alias_case, 39–55
 alias_database_url, 39–202
 alias_domains, 39–56
 Compared with aliaswild channel option,
 33–31
 alias_entry_cache_negative, 39–152
 alias_entry_cache_size, 39–152
 alias_entry_cache_timeout, 39–152
 Lag in seeing LDAP alias changes take effect,
 35–48
 alias_hash_size, 39–176
 alias_magic, 39–56
 Aliases in LDAP, 35–5
 Override via \$nT rewrite rule control
 sequence, 34–32
 alias_member_size, 39–176
 alias_url0
 Example, 35–7
 ims-ms channels, 51–3, 51–4
 alias_urlN, 39–85
 Aliases in LDAP, 35–5
 Alternative to alias file LDAP URL alias
 values, 35–41
 allow_unquoted_addrs_violate_rfc2798, 39–92
 ap_debug, 39–73
 Archiving
 message_hash_fields, 39–204
 unique_id_template, 39–205
 autoreply_timeout_default, 39–65
 Autoresponse periodicity, 39–64
 Vacation message not generated, 5–45
 vacation_cleanup, 39–66
 vacation_maximum_timeout, 39–66, 39–102
 vacation_minimum_timeout, 39–66, 39–102
 vacation_template, 39–67
 blocked_mail_from_ips, 39–155
 block_limit, 33–112, 39–205
 block_size, 39–206
 alias_blocklimit alias option, 35–11
 [BLOCKLIMIT] alias file named parameter,
 35–30
 bounce_block_limit, 39–206, 39–213
 Notification messages, 47–25
 buffer_size, 39–171
 BURL, 39–68
 imap_password, 39–68
 imap_username, 39–68
 cache_debug, 39–73
 cache_magic -- OBSOLETE, 39–171
 capture_format_default, 39–92
 ldap_capture MTA option, 39–117
 Changes take effect, 39–10
 chunk_cache_limit, 39–177
 circuitcheck_completed_bins, 39–69
 circuitcheck_paths_size, 39–177
 comment_chars, 39–171
 Alias database, 35–44
 Domain database, 34–35
 General database, 37–21
 configutil parameter override , 39–68
 config_debug, 39–73
 content_return_block_limit, 39–207, 39–213
 Notification messages, 47–25
 conversions, 38–1, 39–69
 Conversions, 39–69
 conversion_size, 39–177
 Counters, 39–69
 circuitcheck_completed_bins, 39–70
 enable_delay_timers, 39–70
 Databases, 39–71
 alias_database_url, 39–202
 domain_database_url, 39–203
 forward_database_url, 39–203
 general_database_url, 39–203

- name_table_name (OpenVMS only), 39–59
- reverse_database_url, 39–203
- reverse_data_size, 39–180
- ssr_database_url, 39–203
- use_alias_database, 39–60
- use_domain_database, 39–61
- use_forward_database, 39–61, 39–199
- use_reverse_database, 39–62, 39–200
- use_text_databases, 39–175
- Debug, 39–71
 - ap_debug, 39–73
 - cache_debug, 39–73
 - config_debug, 39–73
 - debug_flush, 39–73, 39–172
 - dequeue_debug, 39–73
 - filter_debug, 39–74, 39–231
 - log_debug, 39–74
 - mm_debug, 39–74
 - os_debug, 39–74
 - post_debug, 39–75
 - return_debug, 39–75
 - tracking_debug, 39–75
- debug_flush, 39–73, 39–172
 - Dispatcher debugging, 39–73, 39–172
- decode_encoded_words, 39–224
- defer_group_processing, 39–184
 - Mass mailings, 36–21
- defer_header_addition, 39–224
 - Sieve redirect action, 5–41
- delimiter_char, 39–57
- delivery_options, 39–92
 - autoreply (vacation) for mailing lists and groups, 39–93
 - Comment confusion, 39–95
 - Custom clauses for custom ims-ms_* channel delivery, 51–5
 - Effect on mailRoutingHosts interpretation, 39–144
 - Forwarding user's mail, 35–59
 - Group default delivery approach, 39–95
 - Hold channel, 52–10
 - LMTP, 39–94
 - mailbox delivery for ims-ms channel, 51–3
 - nomail clause, 39–94
 - Order of clauses, 39–95
 - Pipe channel, 52–14
 - preserve subaddress in held messages, 39–94
 - User default delivery approach, 39–95
- dequeue_debug, 39–73
 - os_debug MTA option, 39–75
- dequeue_map, 39–172
- describe_cache_limit, 39–177
- digest_on, 39–184
- Direct LDAP alias lookups, 35–5
- Direct LDAP attribute interpretation, 39–91
 - aliasdetourhost_null_optin, 39–91
 - delivery_options, 39–92
 - group_dn_template, 39–96
 - ldap_host_alias_list, 39–84, 39–98
 - ldap_local_host, 39–84, 39–98
 - ldap_uid_invalid_chars, 39–99
 - process_substitutions, 39–99
 - route_to_routing_host, 39–101
 - spare_*_separator, 39–101
 - vacation_maximum_timeout, 39–66, 39–66, 39–102, 39–102
- Direct LDAP attribute names, 39–102
 - Capture trigger, 39–116, 39–148
 - ldap_add_header, 39–138
 - ldap_alias_addresses, 39–121
 - ldap_attr_domain1_schema2, 39–81, 39–142
 - ldap_attr_domain2_schema2, 39–82, 39–142
 - ldap_attr_domain_search_filter, 39–82, 39–88, 39–142
 - ldap_auth_domain, 39–133
 - ldap_auth_mappingN, 39–141
 - ldap_auth_password, 39–133
 - ldap_auth_policy, 39–132
 - ldap_auth_url, 39–132
 - ldap_autoreply_addresses, 39–128
 - ldap_autoreply_mode, 39–126
 - ldap_autoreply_subject, 39–126
 - ldap_autoreply_text, 39–127
 - ldap_autoreply_text_internal, 39–128
 - ldap_autoreply_timeout, 39–128
 - ldap_autosecretary, 39–122
 - ldap_blocklimit, 39–123
 - ldap_cant_domain, 39–132
 - ldap_cant_url, 39–132
 - ldap_check_header, 39–141
 - ldap_conversion_tag, 39–123
 - ldap_creation_date, 39–151
 - ldap_delay_notifications, 39–138
 - ldap_delivery_file, 39–125
 - ldap_delivery_option, 39–120
 - ldap_detourhost_optin, 39–123
 - ldap_digest_interval, 39–138
 - ldap_disk_quota, 39–124
 - ldap_domain_attr_alias, 7–6, 39–143
 - ldap_domain_attr_autoreply_timeout, 39–147
 - ldap_domain_attr_autosecretary, 39–146
 - ldap_domain_attr_basedn, 7–6, 39–143
 - ldap_domain_attr_canonical, 39–143
 - ldap_domain_attr_capture, 39–148
 - ldap_domain_attr_catchall_address, 39–149
 - ldap_domain_attr_catchall_mapping, 39–149

- ldap_domain_attr_conversion_tag, 39–146
- ldap_domain_attr_creation_date, 39–151
- ldap_domain_attr_default_mailhost, 39–147
- ldap_domain_attr_detourhostoptin, 39–150
- ldap_domain_attr_disk_quota, 39–147
- ldap_domain_attr_filter, 39–148
- ldap_domain_attr_filter, Sieve hierarchy, 5–65
- ldap_domain_attr_mail_status, 7–7, 39–145
- ldap_domain_attr_message_quota, 39–148
- ldap_domain_attr_nosolicit, 39–147
- ldap_domain_attr_optinN, 39–146
- ldap_domain_attr_presence, 39–146
- ldap_domain_attr_recipientcutoff, 39–150
- ldap_domain_attr_recipientlimit, 39–150
- ldap_domain_attr_report_address, 39–148
- ldap_domain_attr_routing_hosts, 39–144
- ldap_domain_attr_smarthost, 39–144
- ldap_domain_attr_sourceblocklimit, 39–149
- ldap_domain_attr_source_channel, 39–150
- ldap_domain_attr_source_conversion_tag, 39–146
- ldap_domain_attr_status, 7–7, 39–145
- ldap_domain_attr_subaddress, 39–144
- ldap_domain_attr_uid_separator, 7–6, 39–144
- ldap_domain_attr_uplevel, 39–143
- ldap_end_date, 39–123
- ldap_equivalence_addresses, 39–121
- ldap_expandable, 39–140
- ldap_filter, 39–129
- ldap_filter, Sieve hierarchy, 5–65
- ldap_filter_reference, 39–129
- ldap_filter_reference, Sieve hierarchy, 5–65
- ldap_forwarding_address, 39–130
- ldap_group_dn, 39–135
- ldap_group_dn2, 39–136
- ldap_group_last_access_time, 39–134
- ldap_group_mail_status, 39–115
- ldap_group_rfc822, 39–137
- ldap_group_status, 39–115
- ldap_group_url1, 39–135
- ldap_group_url2, 39–135
- ldap_jettison_domain, 39–130
- ldap_jettison_url, 39–131
- ldap_list_id, 39–131
- ldap_mailhost, 39–124
- ldap_maximum_messages_per_day, 39–133
- ldap_maximum_message_size, 39–133
- ldap_message_quota, 39–125, 39–125
- ldap_mlsrange, 39–116
- ldap_moderator_url, 39–134
- ldap_nosolicit, 39–119
- ldap_objectclass, 39–114
- ldap_optin*, 39–121
- ldap_parental_controls, 39–129
- ldap_personal_name, 39–120
- ldap_preferred_country, 39–119
- ldap_preferred_language, 39–118
- ldap_prefix_text, 39–140
- ldap_presence, 39–122
- ldap_primary_address, 39–120
- ldap_recipientcutoff, 39–117
- ldap_recipientlimit, 39–117
- ldap_reject_action, 39–131
- ldap_reject_text, 39–131
- ldap_remove_header, 39–139
- ldap_reprocess, 39–130
- ldap_routing_address, 39–119
- ldap_sourceblocklimit, 39–118
- ldap_source_channel, 39–118
- ldap_source_conversion_tag, 39–120
- ldap_source_optinN, 39–118
- ldap_start_date, 39–122
- ldap_suffix_text, 39–140
- ldap_url_result_mapping, 39–137
- ldap_user_mail_status, 39–114
- ldap_user_status, 39–114
- uid, 39–115
- Direct LDAP domain lookup, 34–29, 39–78
- Direct LDAP lookup cache
 - ldap_domain_timeout, 7–5, 39–83, 39–153
- Direct LDAP schema, 39–88
 - ldap_basedn_filter_schema1, 7–7, 39–82, 39–88
 - ldap_basedn_filter_schema2, 7–7, 39–82, 39–89
 - ldap_domain_filter_schema1, 7–8, 39–83, 39–89
 - ldap_domain_filter_schema2, 7–8, 39–83, 39–89
 - ldap_domain_root, 39–83, 39–89
 - ldap_global_config_templates, 39–89
 - ldap_group_object_classes, 39–89
 - ldap_schematag, 39–90
 - ldap_user_object_classes, 39–90
 - ldap_user_root, 39–87, 39–91
- Direct LDAP user/group lookup, 39–84
 - alias_urlN, 39–85
 - allow_unquoted_addrs_violate_rfc2798, 39–92
 - ldap_default_attr, 39–86
 - ldap_mail_aliases, 39–87
 - ldap_mail_reverses, 39–87
 - reverse_url, 39–88
- Directory location, 39–154
 - langdir, 39–154
 - tmpdir, 39–154

dis_nesting, 39–271
 DKIM, 39–154

- dkim_ignore_domains, 39–154
- dkim_ignore_domains, dkimpreserve interaction, 33–56
- dkim_ignore_domains, dkimremove interaction, 33–56
- dkim_ignore_domains, dkim_preserve_domains interaction, 39–155
- dkim_ignore_domains, dkim_remove_domains interaction, 39–155
- dkim_preserve_domains, 39–155
- dkim_preserve_domains, dkimpreserve interaction, 33–56
- dkim_remove_domains, 39–155
- dkim_remove_domains, dkimremove interaction, 33–56

 dkim_ignore_domains, 39–154

- dkimpreserve interaction, 33–56
- dkimremove interaction, 33–56
- dkim_preserve_domains interaction, 39–155
- dkim_remove_domains interaction, 39–155

 dkim_preserve_domains, 39–155

- dkimpreserve interaction, 33–56

 dkim_remove_domains, 39–155

- dkimremove interaction, 33–56

 DNS lookups, 39–155
 domain_database_url, 39–203
 domain_failure, 39–79

- Direct LDAP domain lookups, 34–30, 34–30

 domain_hash_size, 39–178
 domain_match_cache_size, 39–152

- Direct LDAP domain lookups, 34–30, 34–30

 domain_match_cache_timeout

- Direct LDAP domain lookups, 34–30, 34–30

 domain_match_url, 39–80

- Direct LDAP domain lookups, 34–30, 34–30
- Example, 35–8

 domain_uplevel, 39–81

- Direct LDAP domain lookups, 34–30, 34–30
- Example, 35–7

 duplicate_maximum_timeout, 39–230
 duplicate_minimum_timeout, 39–230
 duplicate_timeout_default, 39–230
 duplicate_tracking_url, 39–230
 enable, 39–54

- Default for schedule.task:purge.enable, 8–3, 16–24
- Default for schedule.task:return_job.enable, 8–3, 8–4, 8–4

 enable_delay_timers, 39–70
 enable_sieve_body, 5–22, 39–228
 enable_sieve_ereject, 39–228
 enable_sieve_memcache, 39–228

- Disabling memcache Sieve extension, 5–53

 enable_sieve_metermaid, 39–229

- Disabling metermaid Sieve extension, 5–57

 enable_sieve_regex, 39–229

- regex Sieve extension, 5–61

 Error interpretation, 39–156

- access_errors, 39–156
- spamfilterN_optional, 39–250
- use_permanent_error, 39–168
- use_temporary_error, 39–169

 Error text, 39–156

- error_text_wrong_account, checkrrvs channel option, 33–36, 33–119
- error_text_wrong_domain, checkrrvs channel option, 33–36, 33–119

 error_text_*, 39–157
 error_text_accepted_return_address, 39–166
 error_text_access_failure, 39–158
 error_text_alias_auth, 39–158
 error_text_alias_fileerror, 39–158

- Effect of use_permanent_error MTA option, 39–169

 error_text_alias_fileexist, 39–158

- Effect of use_permanent_error MTA option, 39–169

 error_text_alias_locked, 39–158
 error_text_alias_temp, 39–158
 error_text_block_over, 39–159
 error_text_deleted_group, 39–162
 error_text_deleted_user, 39–162
 error_text_disabled_alias, 39–161

- Effect of use_temporary_error MTA option, 39–169

 error_text_disabled_group, 39–162
 error_text_disabled_user, 39–161

- Effect of use_temporary_error MTA option, 39–169

 error_text_inactive_group

- Effect of use_permanent_error MTA option, 39–169

 error_text_inactive_user

- Effect of use_permanent_error MTA option, 39–169

 error_text_over_quota

- Effect of use_permanent_error MTA option, 39–169

 error_text_recipient_over

- Effect of use_permanent_error MTA option, 39–169
- recipientlimit channel option, 33–87, 33–121

 error_text_spf_*

- spfmailfrom and spfrcptto channel options, 33-145
- error_text_still_held
 - Hold channel, 52-11
- error_text_transaction_limit_exceeded
 - transactionlimit channel option, 33-125
- error_text_unknown_alias
 - Effect of use_temporary_error MTA option, 39-169
- error_text_unknown_host
 - Effect of use_temporary_error MTA option, 39-169
- error_text_unknown_user
 - Effect of use_temporary_error MTA option, 39-170
- error_text_wrong_account, 39-161
 - checkrrvs channel option, 33-36, 33-119
- error_text_wrong_domain, 39-161
 - checkrrvs channel option, 33-36, 33-119
- expandable_default, 39-184
- exproute_forward, 39-57
 - exproute channel option, 33-38
- External filtering context, 39-170
 - scan_channel, 39-170
 - scan_originator, 39-170
 - scan_recipient, 39-170
- External LDAP directory
 - ldap_ext_host, 39-182
 - ldap_ext_max_connections, 39-182
 - ldap_ext_password, 39-182
 - ldap_ext_port, 39-182
 - ldap_ext_username, 39-182
- fdirectory, 39-172
- File format and file handling, 39-171
- File format and handling
 - cache_magic -- OBSOLETE, 39-171
 - comment_chars, 39-171
 - dequeue_map, 39-172
 - fdirectory, 39-172
 - fsync, 39-172
 - log_alq, 39-173
 - log_deq, 39-173
 - max_internal_blocks, 39-173
 - mm_abc, 39-173
 - mm_mbf, 39-173
 - osync, 39-174
 - queue_cache_mode, 39-174
 - queue_cache_mode_3_files, 39-174
 - use_text_databases, 39-174
- file_member_size, 39-178
- filter_cache_size, 39-228
- filter_cache_timeout, 39-228
- filter_debug, 39-74, 39-231
- filter_discard, 39-224
 - filter_discard channel, 52-7
- filter_jettison, 39-224
 - filter_discard channel, 52-7
- form_names, 39-272
- FORWARD mapping tables
 - include_spares2, 39-196
- forward_database_url, 39-203
- forward_data_size, 39-178
- fruits_size, 39-178
- fsync, 39-172
- general_database_url, 39-203
- general_data_size, 39-178
 - General database, 37-21
- group_dn_template, 39-96, 39-96
 - Mailing list membership, 36-10
- header_limit, 39-207
 - headerlimit channel option, 33-71
- held_sndopr, 39-220, 39-246
 - Diagnosing .HELD files, 52-11
- history_to_return, 39-213
 - return_delivery_history MTA option, 39-215
- host_hash_size, 39-179
- id_domain, 39-221
 - Limiting emission of internal host names, 57-2
 - Local channel official_host_name, 52-2
- imap_password, 39-68
 - BURL usage, 49-11
- imap_username, 39-68
 - BURL usage, 49-11
- improute_forward, 39-57
 - improute channel option, 33-39
- include_connectioninfo, 39-189
- include_conversiontag, 39-190
 - *_ACCESS mapping table probes, 44-7
 - tag switch of test -rewrite utility, 58-38
 - CHARSET-CONVERSION mapping table, 38-17
 - FORWARD mapping table probes, 35-59
 - FROM_ACCESS mapping table, 44-14
- include_mtpriority, 39-191
 - FORWARD mapping table probes, 35-59
- include_spares, 39-192
 - *_ACCESS mapping table probes, 44-7
 - FROM_ACCESS mapping table, 44-14
- include_spares1, 39-193
 - *_ACCESS mapping table probes, 44-7
- include_spares2, 39-197
 - FORWARD mapping table probes, 35-59
 - ldap_spare_N values, 39-125
- Internal size, 39-175
 - alias_hash_size, 39-176

- alias_member_size, 39-176
- circuitcheck_paths_size, 39-177
- conversion_size, 39-177
- describe_cache_limit, 39-177
- domain_hash_size, 39-178
- file_member_size, 39-178
- forward_data_size, 39-178
- fruits_size, 39-178
- general_data_size, 39-178
- host_hash_size, 39-179
- ldap_attr_name_hash_size, 39-179
- ldap_object_class_hash_size, 39-179
- map_names_size, 39-180, 39-197
- options_hash_size, 39-180
- personal_conversion_size, 39-180
- reverse_data_size, 39-180
- wild_pool_size, 39-181
- journal_format, 39-96, 39-204
 - "capture :journal" message copies, 54-16
- langdir, 39-154
 - Fallback location for return_*.txt files, 47-10
- LDAP and URL lookup cache, 39-152
- LDAP attributes returned upon authentication
 - ldap_auth_attr_mail_host, 39-151
 - ldap_auth_attr_sender, 39-151
 - ldap_auth_attr_submit_channel, 39-151
- LDAP attributes returned with authentication
 - ldap_auth_attr_recall_secret, 39-152
- LDAP bind and connect, 39-76
 - ldap_host, 39-76
 - ldap_max_connections, 39-76
 - ldap_password, 39-76
 - ldap_port, 39-77
 - ldap_timeout, 39-77
 - ldap_username, 39-78
 - ldap_use_async, 39-77
 - max_urls, 39-78
- LDAP external directory, 39-181
- LDAP lookup cache
 - cache_debug, 39-73
 - reverse_address_cache_size, 39-153
 - reverse_address_cache_timeout, 39-153
 - url_result_cache_size, 39-154
- LDAP PAB, 39-182
- ldap_add_header, 39-139
- ldap_alias_addresses, 39-121
 - Address reversal, 35-49
- ldap_alternate_recipient, 39-122
- ldap_attr_domain1_schema2, 39-81, 39-142
- ldap_attr_domain2_schema2, 39-82, 39-142
- ldap_attr_domain_search_filter, 39-82, 39-88, 39-142
- ldap_attr_name_hash_size, 39-179
- ldap_auth_attr_mail_host, 39-151
- ldap_auth_attr_recall_secret, 39-152
- ldap_auth_attr_sender, 39-151
- ldap_auth_attr_submit_channel, 33-20, 39-151
 - Use with FUTURERELEASE, 49-12
- ldap_auth_domain, 39-133
- ldap_auth_mappingN, 39-141
- ldap_auth_password, 39-133
- ldap_auth_policy, 39-132
- ldap_auth_url, 39-132
 - process_substitutions MTA option, 39-100
- ldap_autoreply_addresses, 39-128
 - Vacation message not generated, 5-45
- ldap_autoreply_mode, 39-126
- ldap_autoreply_subject, 39-126
- ldap_autoreply_text, 39-127
 - Vacation message not generated, 5-46
- ldap_autoreply_text_internal, 39-127
 - Vacation message not generated, 5-46
 - vnd.sun.autoreply-internal Sieve environment item, 5-15
- ldap_autoreply_timeout, 39-65, 39-128
 - Vacation message not generated, 5-45
 - vacation_maximum_timeout MTA option, 39-66, 39-102
 - vacation_minimum_timeout MTA option, 39-67, 39-102
- ldap_autosecretary, 39-122
- ldap_basedn_filter_schema1, 7-7, 39-82, 39-89
- ldap_basedn_filter_schema2, 7-7, 39-82, 39-89
- ldap_blocklimit, 33-112, 39-123
 - Address reversal, 35-49
 - Notification messages, 47-25
- ldap_cant_domain, 39-132
- ldap_cant_url, 39-132
 - process_substitutions MTA option, 39-100
- ldap_capture, 39-116
 - Address reversal, 35-49
- ldap_check_header, 39-141
- ldap_conversion_tag, 39-123
- ldap_creation_date, 39-151
- ldap_default_attr, 39-86
- ldap_default_domain, 39-82, 39-97
 - Direct LDAP alias lookups, 35-6
 - Direct LDAP domain lookups, 34-30
 - ims-ms channels, 51-4
 - Twin of base.defaultdomain, 7-5, 27-13
- ldap_delay_notifications, 39-138
- ldap_delivery_file, 39-125
- ldap_delivery_option, 39-120
 - Deferred expansion of groups, 39-184
 - delivery_options interpretation, 39-92
 - Forwarding user's mail, 35-59

- ims-ms channels, 51–4
- ldap_detourhost_optin, 39–123
 - aliasoptindetourhost_null_optin specifies ignored value of, 39–91
- ldap_digest_interval, 39–138
- ldap_disk_quota, 39–124
 - User LDAP attribute to override defaultmailboxquota, 16–11
- ldap_domain_attr_alias, 7–6, 39–143
- ldap_domain_attr_autoreply_timeout, 39–65, 39–147
- ldap_domain_attr_autosecretary, 39–146
- ldap_domain_attr_basedn, 7–6, 39–142
- ldap_domain_attr_blocklimit, 33–112
 - Address reversal, 35–50
- ldap_domain_attr_canonical, 39–143
- ldap_domain_attr_capture, 39–148
- ldap_domain_attr_catchall_address, 39–149
- ldap_domain_attr_catchall_mapping, 39–149
- ldap_domain_attr_conversion_tag, 39–146
- ldap_domain_attr_creation_date, 39–151
- ldap_domain_attr_default_mailhost, 39–147
- ldap_domain_attr_detourhostoptin, 39–150
- ldap_domain_attr_disk_quota, 39–147
- ldap_domain_attr_filter, 39–148
 - Sieve hierarchy, 5–65
- ldap_domain_attr_mailserv, 39–143
- ldap_domain_attr_mail_status
 - Hold channel, 52–10
 - Hold channel, Releasing messages, 52–11
- ldap_domain_attr_message_quota, 39–147
- ldap_domain_attr_nosolicit, 39–147
- ldap_domain_attr_optinN, 39–146
- ldap_domain_attr_presence, 39–146
- ldap_domain_attr_recipientcutoff, 39–150
 - Address reversal, 35–50
 - Compared to channel options, 33–88, 33–121
- ldap_domain_attr_recipientlimit, 39–150
 - Address reversal, 35–50
 - Compared to channel options, 33–88, 33–121
- ldap_domain_attr_report_address, 39–148
 - Address reversal, 35–50
- ldap_domain_attr_routing_hosts, 39–144
 - Interpretation affected by route_to_routing_host MTA option, 39–101
 - Routing to a gateway system, 49–47
- ldap_domain_attr_smarthost, 39–144
 - Routing to a gateway system, 49–47
- ldap_domain_attr_sourceblocklimit, 33–112, 39–149
 - Address reversal, 35–50
- ldap_domain_attr_source_channel, 39–150
 - Address reversal, 35–50
- ldap_domain_attr_source_conversion_tag, 39–146
 - Address reversal, 35–50
 - Subaddresses and LDAP lookups, 35–45
- ldap_domain_attr_uid_separator, 7–6, 39–144
- ldap_domain_attr_uplevel, 39–143
- ldap_domain_filter_schema1, 7–8, 39–83, 39–89
 - Direct LDAP domain lookups, 34–30
- ldap_domain_filter_schema2, 7–8, 39–83, 39–89
 - Direct LDAP domain lookups, 34–30
- ldap_domain_known_attributes, 7–5, 39–83
 - Direct LDAP domain lookups, 34–30, 34–30
- ldap_domain_root, 39–83, 39–89
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
 - Twin of base.dccroot, 7–16
- ldap_domain_timeout, 7–5, 39–83, 39–153
 - Direct LDAP domain lookups, 34–30, 34–30
 - TCP wrappers, 6–2
- ldap_end_date, 39–123
 - Current date comparison for vacation message generation, 5–45
- ldap_equivalence_addresses, 39–121
 - Address reversal, 35–49
- ldap_errors_to, 39–137
- ldap_expandable, 39–140
 - expn* channel options, 33–126
- ldap_ext_host, 39–182
- ldap_ext_max_connections, 39–182
- ldap_ext_password, 39–182
- ldap_ext_port, 39–182
- ldap_ext_username, 39–182
- ldap_filter, 39–129
 - Sieve hierarchy, 5–65
- ldap_filter_reference, 39–129
 - Sieve hierarchy, 5–65
- ldap_forwarding_address, 39–130
 - Forwarding user's mail, 35–59
- ldap_global_config_templates, 39–89
- ldap_group_dn, 39–135
 - Interpretation affected by group_dn_template MTA option, 39–96
- ldap_group_dn2, 39–136
 - Interpretation affected by group_dn_template MTA option, 39–96
- ldap_group_last_access_time, 39–134
- ldap_group_mail_status, 39–115
- ldap_group_object_classes, 39–89
 - Direct LDAP alias lookups, 35–6

- ldap_group_rfc822, 39–137
- ldap_group_status, 39–115
 - Supported values, 39–115
- ldap_group_url1, 39–134
 - process_substitutions MTA option, 39–100
- ldap_group_url2, 39–135
 - process_substitutions MTA option, 39–100
- ldap_hoh_filter, 39–97, 39–141
- ldap_hoh_owner, 39–97, 39–141
 - Sieve syntax error notification messages, 47–2
- ldap_host, 39–76
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - Mapping table `$(ldap-url)` substitutions, 37–14
 - Twin of `ugldaphost` base option, 7–14
- ldap_host_alias_list, 39–84, 39–98
 - Direct LDAP alias lookups, 35–6
- ldap_jettison_domain, 39–131
- ldap_jettison_url, 39–131
- ldap_list_id, 39–131
- ldap_local_host, 39–84, 39–98
 - Direct LDAP alias lookups, 35–6
 - L channel `official_host_name`, 33–79
 - Twin of `base.hostname`, 7–9
- ldap_mailhost, 39–124
 - aliasdetourhost interaction, 33–30, 33–60
- ldap_mail_aliases, 39–86
 - Adjusting when `ldap_equivalence_address` is changed, 39–121
 - Direct LDAP alias lookups, 35–6
- ldap_mail_reverses, 39–87
- ldap_maximum_messages_per_day, 39–133
- ldap_maximum_message_size, 33–112, 39–133
- ldap_max_connections, 39–76
 - Direct LDAP domain lookups, 34–30, 35–6
- ldap_message_quota, 39–125, 39–125
 - User LDAP attribute to override `defaultmessagequota`, 16–11
- ldap_mlsrange, 39–116
- ldap_moderator_url, 39–134
 - process_substitutions MTA option, 39–100
- ldap_nosolicit, 39–119
- ldap_objectclass, 39–114
- ldap_object_class_hash_size, 39–179
- ldap_optin*, 39–121
- ldap_pab_host, 39–183
- ldap_pab_max_connections, 39–183
- ldap_pab_password, 39–183
- ldap_pab_port, 39–183
- ldap_pab_username, 39–183
- ldap_parental_controls, 39–129
- ldap_password, 39–76
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - Twin of `base.ugldapbindcred`, 7–14
- ldap_personal_name, 39–120
 - Address reversal, 35–49
 - PERSONAL_NAMES mapping table, 35–55
- ldap_port, 39–77
 - Default for `ldap_ext_port` MTA option, 39–182
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - Mapping table `$(ldap-url)` substitutions, 37–14
 - Twin of `ugldapport` base option, 7–15
- ldap_preferred_country, 39–119
 - Address reversal, 35–49
- ldap_preferred_language, 39–118
 - Address reversal, 35–49
- ldap_prefix_text, 39–140
- ldap_presence, 39–122
- ldap_primary_address, 39–120
 - Address reversal, 35–49
- ldap_program_info
 - `$P` substitution in LDAP URLs, 1–7
- ldap_recipientcutoff, 39–117
 - Address reversal, 35–49
 - Compared to channel options, 33–88, 33–121
- ldap_recipientlimit, 39–117
 - Address reversal, 35–49
 - Compared to channel options, 33–88, 33–121
- ldap_reject_action, 39–131
- ldap_reject_text, 39–131
- ldap_remove_header, 39–139
- ldap_reprocess, 39–130
 - Deferred expansion of groups, 39–184
 - Mass mailings, 36–21
- ldap_routing_address, 39–119
- ldap_schemalevel, 7–5, 39–90
- ldap_schematag, 39–90
 - Affects default for `ldap_alias_addresses` MTA option, 39–121
 - Direct LDAP alias lookups, 35–6
 - Direct LDAP domain lookups, 34–30
- ldap_sourceblocklimit, 33–112, 39–118
 - Address reversal, 35–49
- ldap_source_channel, 39–118
 - Address reversal, 35–49
 - Name of attribute used for `userswitchchannel` purposes, 33–82
 - `userswitchchannel` channel option, 33–20
- ldap_source_conversion_tag, 39–120
 - Address reversal, 35–49
- ldap_source_optin

- Address reversal, 35–49
- ldap_source_optin*
 - Archiving, 54–21
- ldap_source_optinN, 39–118
- ldap_spare_1
 - Address reversal, 35–49
- ldap_spare_2
 - Address reversal, 35–49
- ldap_spare_3
 - Address reversal, 35–49
- ldap_spare_4
 - Address reversal, 35–49
 - Example for Sieve external list, 5–35
 - SIEVE_EXTLISTS mapping probes, 5–30
- ldap_spare_5
 - Address reversal, 35–49
 - SIEVE_EXTLISTS mapping probes, 5–30
- ldap_spare_6
 - SIEVE_EXTLISTS mapping probes, 5–30
- ldap_start_date, 39–122
 - Current date comparison for vacation message generation, 5–45
- ldap_suffix_text, 39–140
- ldap_timeout, 39–77
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
- ldap_uid, 39–115
 - \$M substitution in LDAP URLs, 1–7
- ldap_uid_invalid_chars, 39–99
- ldap_url_result_mapping, 39–137
 - Example, 36–13
 - Relationship to process_substitutions MTA option, 39–100
- ldap_username, 39–78
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - Twin of base.ugldapbinddn, 7–14
- ldap_user_mail_status, 39–114
 - Hold channel, 52–10
 - Hold channel, Releasing messages, 52–11
- ldap_user_object_classes, 39–90
 - Direct LDAP alias lookups, 35–6
- ldap_user_root, 39–87, 39–91
 - Direct LDAP alias lookups, 35–6
 - Twin of base.ugldapbasedn, 7–14
- ldap_user_status, 39–114
- ldap_use_async, 39–77
 - Mass mailings, 36–21
- lines_to_return, 39–214
- line_limit, 33–112, 39–207
- Listed alphabetically, 39–11
- Listed by functional group, 39–24
- local_format_restrictions, 39–57
- local_quota_checks
 - RESTRICTED, 39–207
- Logging
 - log_alternate_recipient, 39–252
 - log_auth, 39–252
 - log_callout_delays, 39–252
 - log_conversion_tag, 39–255
 - log_debug, 39–74
 - log_delivery_flags, 39–262
 - log_deliver_by, 39–255
 - log_filter, systemfilter MTA option, 39–223
 - log_futurerelease, 39–261
 - log_imap_flags, 39–262
 - log_mailbox_uid, 39–263
 - log_mtpriority, 39–264
 - log_times, 39–268
 - log_tracking, 39–268
 - log_transactionlog, 39–232, 39–268
 - log_uid, 39–269
 - log_username, 39–269
 - log_use_xtext, 39–269
- Logging, monitoring, and counters, 39–251
- loglevel, 39–72, 49–16
- log_alq, 39–173, 39–251
- log_alternate_recipient, 39–252
- log_auth, 39–252
- log_callout_delays, 39–252
- log_connection, 39–254, 55–1
 - \$T flag in PORT_ACCESS mapping table, 44–4
 - Example, 55–4
- log_connections_syslog, 39–247
- log_conversion_tag, 39–255
- log_debug, 39–74
- log_delay_bins, 39–70
- log_delivery_flags, 39–262
- log_deliver_by, 39–255
- log_deq, 39–173, 39–251
- log_diagnostics, 39–256
- log_envelope_id, 39–256, 55–1
 - Example, 55–4
- log_filename, 39–256
 - Example, 55–4
- log_filename_id, 55–1
- log_filter, 39–231, 39–257
 - addprefix or addsuffix actions, 5–49
 - Diagnosing .HELD files, 52–11
 - discard or jettison strings, 5–23
 - Example, 55–4
 - Memcache protocol errors, 5–63
 - Sieve duplicate errors, 5–25, 5–63
 - Sieve vacation errors, 5–44, 5–46, 5–63

Sieve vacation errors, Error <n> modifying key <string> value <string> to vacation file <filename>, 5-44, 5-46

Sieve vacation errors, Error updating vacation memcache entry <errno> <string>, 5-44, 5-46

Sieve vacation errors, Memcache client error adding entry <errno> <string>, 5-44, 5-46

Sieve vacation errors, Memcache server error adding entry <errno> <string>, 5-44, 5-46

Sieve vacation errors, Server error adding memcache vacation entry <errno> <string>, 5-44, 5-46

Sieve vacation errors, Unable to open memcache connection for vacation: <string> (<errno>), 5-44, 5-46

Sieve vacation errors, Vacation file <filename> cannot be opened, 5-44, 5-46

Sieve vacation errors, Vacation operation: Memcache client returned error <errno> <string>, 5-44, 5-46

Sieve vacation errors, Vacation operation: Memcache server returned error <errno> <string>, 5-44, 5-46

spamtest level and virustest level, 5-42

systemfilter MTA option, 39-223

vacation action, 5-44

log_format, 39-257

Influence on defaults for other log_* MTA options, 39-252

log_frustration_limit, 39-70

log_futurerelease, 39-261

log_header, 39-261

Affected by log_messages_syslog, 39-249

Compared with transactionlog use in Sieve script, 39-232, 39-269

log_imap_flags, 39-262

log_intermediate, 39-263

Example, 55-4

log_local, 39-263

log_mailbox_uid, 39-263

log_messages_syslog, 39-247

log_message_id, 39-263, 55-1

Example, 55-4

Notification messages, 47-24

log_mtpriority, 39-264

log_node, 39-264

Example, 55-4

log_notary, 39-265

Example, 55-4

log_priority, 39-265

Example, 55-4

log_process, 39-266, 55-1

Bitbucket channel dequeues, 52-2

Example, 55-4

Notification messages, 47-24

Reprocess channel, 52-19

log_queue_time, 39-266

log_reason, 39-267

log_sensitivity, 39-267

Example, 55-4

log_size_bins, 39-70

log_sndopr, 39-249

log_statistics, 39-70

log_times, 39-268

log_tracking, 39-268

log_transactionlog, 39-232, 39-268

log_uid, 39-269

log_username, 39-269, 55-1

Example, 55-4

filter_discard channel logs as FILTER_DISCARD, 52-9

log_use_xtext, 39-269

Mailing lists and groups, 39-184

expandable_default, 39-184

mail_off, 39-185

or_clauses, 39-185

post_off, 39-185

MAILSERV, 39-186

mailserv_moderator_mail, 39-186

mailserv_moderator_uid, 39-186

mailserv_secret, 39-186

MAILSERV LDAP schema, 39-186

MAILSERV list subscription LDAP attribute names, 39-187

MAILSERV managed lists, 39-187

MAILSERV moderator user, 39-186

MAILSERV user LDAP attribute names, 39-186

mailserv_moderator_mail, 39-186

mailserv_moderator_uid, 39-186

mailserv_secret, 39-186

mail_delivery_filename, 39-272

mail_off, 39-185

Mailing list members, 36-22

Mapping table miscellaneous, 39-196

Mapping tables, 39-188

mapping_paranoia, 39-194

*_ACCESS mapping table probes, 44-8

AUTH_ACCESS mapping table, 49-37

BURL_ACCESS mapping table, 49-8

MILTER_MACROS mapping table, 45-14

map_names_size, 39-180, 39-197

max_addheaders, 5-25, 39-225

max_alias_levels, 39-58

max_duplicates, 39-226, 39-231

max_fileintos, 39-226

max_header_blocks, 39-208

- max_header_block_use, 33–48, 39–208
- max_header_lines, 39–208
- max_header_line_use, 33–48, 39–208
- max_internal_blocks, 39–173
- max_local_received_lines, 39–221
- max_mime_levels, 39–209
 - Diagnosing .HELD files, 52–11
- max_mime_parts, 39–209
 - Diagnosing .HELD files, 52–11
- max_mr_received_lines, 39–221
- max_notifys, 39–226
- max_received_lines, 39–221
- max_redirect
 - Sieve redirect action, 5–40
- max_redirects, 39–226
- max_redirect_addresses, 39–226
 - redirect to external list, 5–41
 - Sieve external list example, 5–32
- max_sieve_list_size, 39–226
- max_sieve_string_size, 39–227
- max_total_received_lines, 39–222
- max_urls, 39–78
- max_vacations, 39–227
 - vacation action, 5–44
 - vacation action, Vacation message not generated, 5–45
- max_variables, 39–227
- max_x400_received_lines, 39–222
- Memcache, 39–202
 - alias_database_url, 39–202
 - domain_database_url, 39–203
 - enable_sieve_memcache, 39–228
 - forward_database_url, 39–203
 - general_database_url, 39–203
 - memcache_expire, 39–202
 - memcache_host, 39–202
 - memcache_port, 39–202
 - memcache_timeout, 39–202
 - reverse_database_url, 39–203
 - ssr_database_url, 39–203
- memcache_expire, 39–202
 - Use with Message Tracking, 48–1
- memcache_host, 39–202
 - check_memcache.so use of, 37–24
 - Effect on duplicate_tracking_url, 39–230
 - Sieve duplicate test, 5–24
 - Sieve filter memcache extension, 5–53
 - Use with Message Tracking, 48–1
- memcache_port, 39–202
 - check_memcache.so use of, 37–24
 - Effect on duplicate_tracking_url, 39–231
 - Sieve duplicate test, 5–24
 - Use with Message Tracking, 48–1
- memcache_timeout, 39–202
- Message archival and hashing, 39–203
 - journal_format, 39–96, 39–204
- Message fragmentation size limits
 - max_header_block_use, 39–208
 - max_header_line_use, 39–208
- Message loop detection and .HELD messages, 39–220, 39–220, 39–246
- Message size, 39–205
 - block_limit, 39–205
 - block_size, 39–206
 - header_limit, 39–207
 - line_limit, 39–207
 - max_header_blocks, 39–208
 - max_header_lines, 39–208
 - max_mime_levels, 39–209
 - max_mime_parts, 39–209
 - non_urgent_block_limit, 39–209, 39–219
 - normal_block_limit, 39–209, 39–219
 - second_class_block_limit, 39–210, 39–220
 - urgent_block_limit, 39–210, 39–220
- Message tracking, 39–210
 - ldap_auth_attr_recall_secret, 39–152
 - log_tracking, 39–268
 - tracking_debug, 39–75
 - tracking_mode, 39–210
 - tracking_retries, 39–210
 - tracking_retry_delay, 39–210
- Message-id: header lines, 39–220
- message_hash_fields, 39–204
 - Message identifier generation (for archiving), 54–19
- message_save_copy_flags, 39–197
 - Conversion tags, 54–5
 - MESSAGE-SAVE-COPY mapping table probe, 54–4
- MeterMaid, 39–211
 - metermaid_backoff, 39–211
 - metermaid_expire, 39–211
 - metermaid_host, 39–211
 - metermaid_port, 39–211
 - metermaid_secret, 39–211
 - metermaid_timeout, 39–212
- metermaid_backoff, 39–211
- metermaid_expire, 39–211
- metermaid_host, 39–211
- metermaid_port, 39–211
- metermaid_secret, 39–211
- metermaid_timeout, 39–212
- Miscellaneous mapping table
 - map_names_size, 39–180, 39–197
 - message_save_copy_flags, 39–198
 - original_channel_probe, 39–198

- use_comment_strings, 39–198
- use_personal_names, 39–201
- wild_pool_size, 39–181
- missing_address, 39–272
- missing_recipient_group_text, 39–58
- missing_recipient_policy, 39–58
- mls, 39–212
- MLS (Multi Layer Security), 39–212
- mm_debug, 39–74
 - \$U flag in address *_ACCESS mapping tables, 44–9, 44–9
 - \$U flag in PORT_ACCESS mapping table, 44–4
 - Sieve debug action, 5–18
- mm_abc, 39–173
- mm_mbf, 39–173
- mtpriority_policy, 39–219
- MTQP, 39–212
 - mtqp_expire, 39–212
 - mtqp_port, 39–212
 - mtqp_timeout, 39–212
- Multi Level Security
 - mls, 39–212
- multinet_mm_exclusive, 39–272
- name_table_name, 39–59
- non_urgent_block_limit, 39–209, 39–219
- normal_block_limit, 39–209, 39–219
- notary_decode, 39–214
- notary_quote, 39–174, 39–214
 - disposition_*.txt files, 47–18
 - Interpretation of content in notification message template files, 47–9
- Notification messages, 39–213
 - history_to_return, 39–213
 - history_to_return, return_delivery_history MTA option, 39–215
 - lines_to_return, 39–214
 - notary_decode, 39–214
 - notary_quote, 39–174, 39–214
 - return_address, 39–215
 - return_cleanup_period, 39–271
 - return_delivery_history, 39–215
 - return_envelope MTA option, 39–155, 39–215
 - return_personal, 39–216
 - return_split_period, 39–270
 - return_units, 39–216
 - return_verify, 39–75
 - use_precedence, 39–217
 - use_warnings_to, 39–217
- notify_previous_response
 - notify_maximum_timeout, 39–65
 - notify_minimum_timeout, 39–66
- notify_ignore_errors, 5–40, 39–225
- notify_maximum_timeout, 39–65
- notify_minimum_timeout, 39–66
- notify_timeout_default, 39–66
- OpenVMS only
 - form_names, 39–272
 - name_table_name, 39–59
- OpenVMS user agent, 39–271
 - dis_nesting, 39–271
 - form_names, 39–272
 - mail_delivery_filename, 39–272
 - missing_address, 39–272
 - multinet_mm_exclusive, 39–272
 - read_receipt_off, 39–272
 - read_receipt_on, 39–272
 - safe_tcl_mode, 39–273
 - use_mail_delivery, 39–273
 - vms_mail_exclusive, 39–273
- optin_user_carryover, 39–99, 39–234
- options_hash_size, 39–180
- original_channel_probe, 39–198
- or_clauses, 39–185
- osync, 39–174
- os_debug, 39–75
 - \$U flag in PORT_ACCESS mapping table, 44–4
- PAB
 - ldap_pab_host, 39–183
 - ldap_pab_max_connections, 39–183
 - ldap_pab_password, 39–183
 - ldap_pab_port, 39–183
 - ldap_pab_username, 39–183
- Password and TLS, 39–217
- personal_conversion_size, 39–180
- plaintextmncipher, 27–19, 39–217
- post_debug, 39–75
- post_off, 39–185
 - Mailing list members, 36–22
- Processing priority, 39–218
- process_substitutions, 39–99
- projectid, 39–174
- queue_cache_mode, 39–174
 - queue_cache_mode_3_files MTA option, 39–174
- queue_cache_mode_3_files, 39–174
- read_receipt_off, 39–272
- read_receipt_on, 39–272
- Received: header lines, 39–220, 39–220, 39–246
 - received_domain MTA option, 39–222
 - received_version, 39–222
- received_domain, 39–222
 - Limiting emission of internal host names, 57–2
 - Local channel official_host_name, 52–2

received_version, 39-222
 returnenvelope
 DNS verification, test -rewrite utility, 58-35
 return_address, 39-215
 test -rewrite utility's -from switch, 58-30
 return_cleanup_period, 39-271
 return_debug, 39-75
 return_delivery_history, 39-215
 return_envelope, 39-155, 39-215
 returnenvelope channel option, 33-99
 return_personal, 39-216
 return_split_period, 39-270
 return_units, 39-216
 Interpretation of *notices channel options, 33-97
 return_verify, 39-75
 reverse_address_cache_size, 39-153
 reverse_address_cache_timeout, 39-153
 reverse_database_url, 39-203
 Address reversal, 35-52
 reverse_data_size, 39-180
 reverse_envelope, 39-60
 reverse_url, 39-88
 mailDomainMsgMaxBlocks effect, 39-146
 route_to_routing_host, 39-101
 Effect on mailRoutingHosts interpretation, 39-144
 safe_tcl_mode, 39-273
 scan_channel, 39-170
 imexpire, 45-17
 scan_originator, 39-170
 imexpire, 45-17
 scan_recipient, 39-170
 imexpire, 45-17
 second_class_block_limit, 39-210, 39-220
 separate_connection_log, 39-270, 55-1
 Sieve filter logging and debugging, 39-231
 Sieve filters, 39-223
 Caching, 39-227
 decode_encoded_words, 39-224
 Duplicate recent messages, 39-229
 enable_sieve_body, 39-228
 enable_sieve_ereject, 39-228
 enable_sieve_memcache, 39-228
 enable_sieve_memcache, Disabling memcache Sieve extension, 5-53
 enable_sieve_metermaid, 39-229
 enable_sieve_metermaid, Disabling metermaid Sieve extension, 5-57
 enable_sieve_regex, 39-229
 enable_sieve_regex, regex Sieve extension, 5-61
 Error text, 39-231
 filter_debug, 39-74, 39-231
 filter_discard, 39-224
 filter_jettison, 39-224
 Interpretation of, 39-223
 Interpretation of, notify_ignore_errors, 39-225
 Language extensions, 39-228
 Language extensions, notify_ignore_errors, 39-225
 max_addheaders, 39-225
 max_duplicates, 39-226, 39-231
 max_fileintos, 39-226
 max_notifys, 39-226
 max_redirects, 39-226
 max_redirect_addresses, 39-226
 max_redirect_addresses, redirect to external list, 5-41
 max_sieve_list_size, 39-226
 max_sieve_string_size, 39-227
 max_vacations, 39-227
 max_variables, 39-227
 See also External filtering context MTA options, 39-170
 sieve_received, 39-225
 sieve_redirect_add_resent, 39-225
 sieve_redirect_add_resent, Default for redirect action, 5-41
 sieve_user_carryover, 39-101, 39-225
 Size limits, 39-225
 strict_require, 39-229
 strict_require, Sieve extensions, 5-18
 sieve_received, 39-225
 sieve_redirect_add_resent, 39-225
 Default for redirect action, 5-41
 sieve_user_carryover, 39-101, 39-225
 Size limits, 39-225
 strict_require, 39-229
 strict_require, Sieve extensions, 5-18
 sieve_received, 39-225
 sieve_redirect_add_resent, 39-225
 Default for redirect action, 5-41
 sieve_user_carryover, 39-101, 39-225
 smtpproxypassword, 27-22, 39-218
 sndopr_priority, 39-249
 Effect on log_sndopr MTA option, 39-249
 held_sndopr MTA option, 39-221, 39-247
 Spam filter, 39-233
 optin_user_carryover, 39-99, 39-234
 scan_channel, 39-170
 scan_originator, 39-170
 scan_recipient, 39-170
 spamfilterN_action_M, 39-235
 spamfilterN_config_file, 39-235
 spamfilterN_final, 39-238
 spamfilterN_includeheaders, 39-238
 spamfilterN_library, 39-234
 spamfilterN_null_action, 39-238
 spamfilterN_null_optin, 39-235
 spamfilterN_optional, 39-250
 spamfilterN_received, 39-239

- spamfilterN_returnpath, 39–239
- spamfilterN_string_action, 39–239
- spamfilterN_verdict_M, 39–235
- Spam/virus filtering
 - See also External filtering context MTA options, 39–170
- spamfilter2_string_action
 - Example, 45–9
- spamfilterN_action_M, 39–235
- spamfilterN_config_file, 39–235
 - Brightmail, 45–2
 - ClamAV, 45–4
 - ICAP, 45–5
 - Milter, 45–5
 - SpamAssassin, 45–7
- spamfilterN_final, 39–238
- spamfilterN_includeheaders, 39–238
- spamfilterN_library, 39–234
- spamfilterN_null_action, 39–238
- spamfilterN_null_optin, 39–235
- spamfilterN_optional, 39–250
- spamfilterN_received, 39–239
- spamfilterN_returnpath, 39–239
- spamfilterN_string_action, 39–239
- spamfilterN_verdict_M, 39–235
- spare_*_separator, 39–101
- SPF, 39–240
 - spf_max_dns_queries, 39–244
 - spf_max_recursion, 39–244
 - spf_max_time, 39–244
 - spf_smtp_status_fail, 39–240
 - spf_smtp_status_fail_all, 39–241
 - spf_smtp_status_permerror, 39–241
 - spf_smtp_status_softfail, 39–242
 - spf_smtp_status_softfail_all, 39–242
 - spf_smtp_status_temperror, 39–243
- spf_max_dns_queries, 39–244
- spf_max_recursion, 39–244
- spf_max_time, 39–244
- spf_smtp_status_fail, 39–240
 - spf* channel options, 33–144
- spf_smtp_status_fail_all, 39–241
 - spf* channel options, 33–144
- spf_smtp_status_permerror, 39–241
 - spf* channel options, 33–143
 - spfmailfrom channel option, 33–144
- spf_smtp_status_softfail, 39–242
 - spf* channel options, 33–145
- spf_smtp_status_softfail_all, 39–242
- spf_smtp_status_temperror, 39–243
 - spf* channel options, 33–143
 - spfmailfrom channel option, 33–144
- SRS, 39–244
 - srs_domain, 39–245
 - srs_maxage, 39–245
 - srs_secrets, 39–245
 - token_char, 39–60, 39–246
- srs_domain, 39–245
- srs_maxage, 39–245
- srs_secrets, 39–245
- token_char, 39–60, 39–246
- srs_domain, 39–245
- srs_maxage, 39–245
 - error_text_srs_timeout MTA option, 39–163
- srs_secrets, 39–245
- sslnicknames, 27–27, 39–218
- ssr_database_url, 39–203
- strict_require, 39–229
 - Sieve extensions, 5–18
- string_pool_size_3
 - General database, 37–21
- string_pool_size_N, 39–181
- subaddress_char, 39–60
- Syslog, 39–246
- systemfilter, 39–223
 - Sieve hierarchy, 5–65
 - Syntax error report message, 47–2
- tmpdir, 39–154
 - Effect on location of MTA database temp files, 40–4
- token_char, 39–60, 39–246
- tracking_debug, 39–75
- tracking_mode, 39–210
- tracking_retries, 39–210
- tracking_retry_delay, 39–210
- Transaction logging
 - log_header, Affected by log_messages_syslog, 39–249
 - return_cleanup_period, 39–271
 - return_split_period, 39–270
- Unified Configuration presentation of, 39–8
- unique_id_template, 39–205
 - Message identifier generation (for archiving), 54–19
- urgent_block_limit, 39–210, 39–220
- url_result_cache_size, 39–154
- user_case, 39–64
- use_alias_database, 39–60
- use_auth_return, 39–194
 - FORWARD mapping table probes, 35–59
 - From address in address-based *_ACCESS mapping table probes, 44–7
 - From address in FROM_ACCESS mapping table probe, 44–14
- use_canonical_return, 39–194
 - FORWARD mapping table probes, 35–59
 - From address in address-based *_ACCESS mapping table probes, 44–7
 - From address in FROM_ACCESS mapping table probe, 44–14

- use_comment_strings, 39–198
- use_domain_database, 39–61
- use_forward_database, 39–61, 39–199
 - Enabling use of forward database, 35–61
 - FORWARD mapping table probes, 35–59
 - FORWARD mapping table probes, Initial and intermediate forms of recipient address, 35–59
- use_ip_access, 39–194
- use_mail_delivery, 39–273
- use_orig_return, 39–194
 - FORWARD mapping table probes, 35–59
 - From address in address-based *_ACCESS mapping table probes, 44–7
 - From address in FROM_ACCESS mapping table probe, 44–14
- use_permanent_error, 39–168
 - recipientlimit channel option, 33–87, 33–121
- use_personal_names, 39–201
- use_precedence, 39–217
- use_reverse_database, 39–62, 39–200
- use_temporary_error, 39–169
- use_text_databases, 39–175
 - Forward database, 35–61
 - Mapping table general database lookups, 37–16
 - Reverse database, 35–52
 - Rewrite rule general database substitutions, 34–24
- use_warnings_to, 39–217, 39–217
- vacation_cleanup, 39–66
- vacation_maximum_timeout, 39–66, 39–102
- vacation_minimum_timeout, 39–66, 39–102
 - Vacation message not generated, 5–45
- vacation_template, 39–67
- Values
 - URL types, 1–4
- vms_mail_exclusive, 39–273
- wild_pool_size, 39–181
- MTA queue directories
 - Reserved for use by Oracle software only, 33–57
- MTA Tailor options, 40–2
 - Directory locations, 40–2
 - File names, 40–4
 - imta_alias_file, 40–6
 - imta_bin, 40–3
 - imta_charset_data, 40–6
 - imta_command_data, 40–5
 - imta_config_data, 40–6
 - imta_config_file, 40–5
 - imta_db_tmp (DELETED), 40–4
 - imta_dl, 40–3
 - imta_general_data (DELETED), 40–7
 - imta_lib, 40–3
 - imta_log, 40–3
 - imta_option_file, 40–5
 - imta_primary_log, 40–6
 - imta_program, 40–4
 - imta_queue, 40–4
 - imta_return_verify
 - Replaced in MS 7.0.5 by return_verify MTA option, 39–75
 - imta_reverse_data (DELETED), 40–7
 - imta_root, 40–2
 - imta_secondary_log, 40–6
 - imta_ssr_database
 - DELETED, 40–8
 - imta_system_filter_file, 40–5
 - imta_table, 40–3
 - imta_tertiary_log, 40–7
 - imta_tmp
 - Effect on location of MTA database temp files, 40–3
 - See tmpdir instead, 40–3
 - imta_user, 40–10
 - imta_user_username, 40–10
 - imta_world_group, 40–10
 - imta_xml_config_file, 40–5
 - Scheduling, 40–10
 - User, 40–10
- MTA user
 - imta_user MTA Tailor option changed to user option in restricted.cnf, 40–10
- mta_channel gateway_profile option, 53–5
- mtindex Message Store options
 - filename, 16–22
 - quotaroot, 16–22
- mtprioritiesallowed channel option, 33–106, 33–130
- mtprioritiesrequired channel option, 33–106, 33–130
- mtpriority_policy MTA option, 39–219
- MTQP
 - MTA options, 39–212
- MTQP (Message Tracking and Query Protocol) server
 - Certificate nickname, 27–27, 39–218
 - Message tracking and recall, 48–1
 - TLS, 27–27, 39–218
- mtqp_expire MTA option, 39–212
- mtqp_port MTA option, 39–212
- mtqp_timeout MTA option, 39–212
- Multi Level Security (MLS)
 - ldap_mlsrange MTA option, 39–116
- multigate channel option, 33–62
 - LMTP, 39–95
- multinet_mm_exclusive MTA option, 39–272

multiple channel option, 33–57
 Channel to a gateway system, 49–47

mustsas channel option, 33–149

mustsasclient channel option, 33–149
 AUTH_ACCESS mapping, 49–38

mustsasserver channel option, 33–149
 Required for implicitsaslexternal to take effect, 33–151
 Should be set on SMTP SUBMIT server channel, 33–119

musttls channel option, 33–83, 33–151

musttlsclient channel option, 33–83, 33–151

musttlsserver channel option, 33–83, 33–151

mx channel option, 33–137
 AUTH_ACCESS mapping table \$M flag, 49–39
 AUTH_ACCESS mapping table \$X flag, 49–39

N

nameparameterlengthlimit channel option, 33–49

nameservers channel option, 33–137
 DNS verification, 33–138
 Reverse lookups, 33–138

Net-SNMP
 Messaging Server's SNMP subagent, 60–1

Network
 Dial up
 ETRN SMTP extension, 49–3, 49–51

newmsg notifytarget option, 24–5

NFS
 Defragment database, 52–3, 52–5
 Vacation response files, 39–67

noaddlineaddrs channel option, 33–29

noaddresssrs channel option, 33–29

noaddreturnpath channel option, 33–63

nobangorpercent channel option, 33–34

nobangoverpercent channel option, 33–34
 Rewrite rule address interpretation, 34–4

nobinaryclient channel option, 33–117

nobinaryserver channel option, 33–117

noblocklimit channel option, 33–112

nocache channel option, 33–135

nochunking* channel options
 BURL interaction, 49–11

nochunkingclient channel option, 33–119

nochunkingserver channel option, 33–119

noconvertoctetstream channel option, 33–45

nodayofweek channel option, 33–66

nodefaulthost channel option, 33–36, 33–67

nodefragment channel option, 33–46

nodeestinationfilter channel option, 33–109

nodestinationsrs channel option, 33–29

nodns channel option, 33–137

nodropblank channel option, 33–68

noehlo channel option, 33–118

noexpirysource channel option, 33–68, 33–105

noexproute channel option, 33–38

nofilecache MSHTTP option, 29–5

nofileinto channel option, 33–110

nofilter channel option, 33–109

noflagtransfer channel option, 33–108, 33–123

noheaderread channel option, 33–70
 Header option file, 33–155

noheadertrim channel option, 33–70
 Header option file, 33–155

noimproute channel option, 33–38

noinner channel option, 33–72

noinnertrim channel option, 33–70
 Header option file, 33–155

nolinelimit channel option, 33–112

nolocalbehavior channel option, 33–39

nologging channel option, 33–84

noloopcheck channel option, 33–128

nomail delivery option, 39–94

nomailfromdnsverify channel option, 33–129, 33–140

nomaster_debug channel option, 33–85

nomsexchange channel option, 33–49, 33–130, 33–152

nomultigate channel option, 33–62

nomx channel option, 33–137

noninbox notifytarget option, 24–5

nonotary channel option, 33–96, 33–131

nonrandommx channel option, 33–137

nonurgentafter channel option, 33–100

nonurgentbackoff channel option, 33–101

nonurgentblocklimit channel option, 33–114
 Job Controller delivery execution window, 42–16

nonurgentnotices channel option, 33–96

nonurgent_delivery Job Controller job_pool option, 42–16

nonurgent_delivery Job Controller option
 Example, 42–6

non_urgent_block_limit MTA option, 39–209, 39–219

noreceivedfor channel option, 33–74
 Limiting emission of internal host names, 57–3

noreceivedfrom channel option, 33–74
 Limiting emission of internal host names, 57–3

noremotehost channel option, 33–36, 33–67

norestricted channel option, 33–40

noreturnaddress channel option, 33–97

noreturnpersonal channel option, 33–97

noreverse channel option, 33–41

normalafter channel option, 33–100

normalbackoff channel option, 33–101

normalblocklimit channel option, 33–114

- Job Controller delivery execution window, 42–6, 42–16
- normalnotices channel option, 33–96
- normal_block_limit MTA option, 39–209, 39–219
- normal_delivery Job Controller job_pool option, 42–16
- norules channel option, 33–41
- nosasl channel option, 33–149
- nosaslclient channel option, 33–149
- nosaslpassauth channel option, 33–154
- nosaslserver channel option, 33–149
- nosaslswitchchannel channel option, 33–82, 33–154
- nosasltrustauth channel option, 33–154
- nosendetrn channel option, 33–131
- nosendpost channel option, 33–94, 47–1
- noserviceconversion channel option, 33–55
- noslave_debug channel option, 33–85
- nosocks channel option, 33–142
- nosourcefilter channel option, 33–109
- nosourcesrs channel option, 33–29
- nosubdirs channel option, 33–59
- noswitchchannel channel option, 33–81
- notary channel option, 33–96, 33–131
- notary_quote MTA option, 39–174, 39–214
 - disposition_*.txt files, 47–18
 - Interpretation of content in notification message template files, 47–9
- nothurman channel option, 33–50
- noticehost alarm option, 11–1
- noticeport alarm option, 11–1
- noticercpt alarm option, 11–1
- notices channel option, 33–96
 - Defragmentation channel, 52–4
 - filter_discard channel, 52–7
 - ims-ms channels, 51–1
 - Local channel value, 52–2
 - Notification message generation, 47–4
 - return_units MTA option, 39–216
- noticesender alarm option, 11–1
 - Postmaster address, 47–2
- notice_time Job Controller option
 - Restricted: for future use, 42–13
- notick channel option, 33–52
- Notification messages, 47–1
 - Channel options, 33–93
 - Delay warnings
 - Mailing list postings, Alias file named parameters, 35–32
 - Format of, 47–4
 - DSN language, 47–8
 - MDN language, 47–17
 - Generation of, 47–3
 - Logging of, 47–24
- Mailing lists
 - alias_keep_delivery alias option, 35–17
 - alias_keep_read alias option, 35–17
 - KEEP_DELIVERY alias file named parameter, 35–35
 - KEEP_READ alias file named parameter, 35–35
- Overquota in Message Store, 47–3
- Postmaster addresses, 47–25
- return_job
 - Scheduler task enable, 8–4
- Routing of, 47–22
- Size limits, 47–25
- Spam bounces, 47–23
- SRS addresses
 - Relay blocking interaction, 49–50
 - Types of, 47–1
- notificationchannel channel option, 33–95, 47–22
- notifytarget options, 24–1
 - annotatmsg, 24–6
 - changeflag, 24–6
 - copymsg, 24–6
 - deletmsg, 24–4
 - enable, 24–2
 - enshost, 24–2
 - Match base.listenaddr option value, 61–1
 - ensport, 24–2
 - Match ens.port option value, 61–1
 - expungemsg, 24–6
 - jmghost, 24–2
 - jmqqport, 24–2
 - jmqqpwd, 24–3
 - jmqqqueue, 24–3
 - jmqttopic, 24–3
 - jmquser, 24–3, 24–4
 - ldapdestination, 24–4
 - loguser, 24–4
 - maxbodysize, 24–3
 - maxheadersize, 24–3
 - msgflags, 24–3
 - msgtypes, 24–5
 - newmsg, 24–4
 - noninbox, 24–5
 - notifytype, 24–2
 - overquota, 24–5
 - persistent, 24–4
 - priority, 24–4
 - purgemsg, 24–5
 - readmsg, 24–5
 - setacl, 24–5
 - ttl, 24–4
 - underquota, 24–5
 - updatemsg, 24–5

- notifytype notifytarget option, 24–2
- notify_ignore_errors MTA option, 5–40, 39–225
- notify_timeout_default MTA option, 39–66
- notls channel option, 33–83, 33–151
- notlscient channel option, 33–83, 33–151
- notlserver channel option, 33–83, 33–151
- nottracking* channel options, 33–91
- notspam MSHTTP feedback option, 29–14
- noturn channel option, 33–132
- novrfy channel option, 33–125
- nowarnpost channel option, 33–94, 47–1
- noxclient channel option, 33–76, 33–132, 33–153
- nox_env_to channel option, 33–75
- numberofhosts PAB option, 59–2
- nummsgs Message Store msghash option, 16–22
- numprocesses IMAP option, 21–12
- numprocesses MMP option, 27–19
- numprocesses MSHTTP option, 29–12
- numprocesses POP option, 22–3
- numthreads MMP option
 - DELETED; see maxthreads, 27–19

O

- obsoleteimap base option, 7–11
- official_host_name channel option, 33–79
 - Domain used to construct message-id's
 - id_domain MTA option overrides, 39–221
 - ims-ms channels, 51–1
 - L channel
 - Postmaster address, 39–215
 - Overridden by ldap_default_domain, 39–222
 - Overridden by local_host_alias, 33–80
 - Overridden by received_domain, 39–222
- On behalf of submission
 - AUTH_ACCESS mapping, 49–39
- OpenDKIM
 - Sieve counters, 5–51
- OpenVMS user agents
 - MTA options, 39–271
- operational Message Store archive option, 16–14
- optin_user_carryover MTA option, 39–99, 39–234
- Options
 - Syntax, 1–1, 39–9
 - Bit-encoded integer, 39–10
 - Boolean, 39–10
 - Floating point, 39–10
 - Integer, Base, 39–10
 - Integer, Bit-encoded, 39–10
 - Special symbolic names, 3–1
- options_hash_size MTA option, 39–180
- or_clauses MTA option, 39–185
 - Effect on alias file named parameter access controls, 35–27
 - Effect on ldap_auth_password processing, 39–134
 - Effect on mailing list access control interactions, 36–2, 36–3
 - Overridden per-list by mgrpBroadcasterPolicy, 39–108
 - Sets default as alias_and vs. alias_or for aliases, 35–10
- os_debug MTA option, 39–75
 - \$U flag in PORT_ACCESS mapping table, 44–4
- overquota notifytarget option, 24–5
- overquotastatus Message Store option, 16–7
 - Enables quota overdraft, 16–8
 - Implies quotaoverdraft, 16–8

P

- PAB, 1
- PAB lookups by the MTA
 - ACI, 39–182
 - MTA options, 39–182
- PAB options, 59–1
 - active, 59–1
 - alwaysusedefaulthost, 59–1
 - attributelist, 59–1
 - defaulthostindex, 59–1
 - enable, 59–1
 - ldapbasedn, 59–1
 - ldapbinddn, 59–2
 - ldaphost, 59–2
 - ldap_pab_host MTA option override for MTA PAB query purposes, 39–183
 - ldappasswd, 59–2
 - ldappport, 59–2
 - ldapusessl, 59–2
 - maxnumberofentries, 59–2
 - migrate415, 59–2
 - numberofhosts, 59–2
- pabldap: URLs
 - MTA URL types, 1–4
- pabldaps: URLs
 - MTA URL types, 1–4
- parameter Dispatcher option, 41–8
- parameterformatdefault channel option, 33–50, 33–54
- parameterformatminimizeencoding channel option, 33–50, 33–54
- parameterformatstripencoding channel option, 33–50, 33–54
- parameterlengthlimit channel option, 33–49
- params pipe option, 52–15
- Parental control
 - ldap_hoh_filter MTA option, 39–97, 39–141
 - ldap_hoh_owner MTA option, 39–97, 39–141

See Sieve filters, Head of household, 5–73

parse_re_* gateway_profile options, 53–6

Partition

- checkdiskusage Message Store option, 16–5
- defaultpartition Message Store option, 16–11

Partition options, 18–1

- cachepath, 18–1, 18–1
- path, 18–1

passyntaxerrors channel option, 33–68

passthrough channel option, 33–119

- destinationdkim* trigger, 33–56
- dkimpreserve trigger, 33–56
- dkim_ignore_domains avoids triggering, 39–154
- dkim_preserve_domains trigger, 39–155

Password

- IMAP ALERT notifications to users warning of expiration, 21–13
- Mailing list postings, 36–3
 - alias_password alias option, 35–20
- On behalf of submission
 - AUTH_ACCESS mapping \$Q flag, 49–38
- SMTP/LMTP client SMTP AUTH use
 - AUTH_ACCESS mapping \$Q flag, 49–38
- User
 - pwchangeurl base option, 7–8
 - Remote password and AUTH_ACCESS mapping, 49–38
 - userPassword LDAP attribute, 39–103

Password expiration

- IMAP ALERT
 - firstwarn pwexpirealert option, 21–13
 - metermaidtable pwexpirealert option, 21–13
 - viametermaid pwexpirealert option, 21–13

path Message Store archive option, 16–15

path partition option, 18–1

Percent signs

- Default for notary_quote MTA option, 39–174, 39–214
- In addresses, 34–4
 - routelocal channel option, 34–8
- In disposition_*.text files, 47–18
- In return_*.txt files, 47–9

percentage Message Store purge option, 16–24

percentonly channel option, 33–34

percents channel option, 33–35

Performance

- File creation
 - osync MTA option, 39–174
- File handling MTA options, 39–171
- LDAP and URL lookup caching
 - MTA options, 39–152
- LDAP lookup caching
 - authcachesize base option, 7–3
 - authcachettl base option, 7–3, 27–6
- LDAP lookups
 - MMP POP and IMAP proxy, 27–27
- LDAP server
 - ldap_domain_known_attributes MTA option, 7–6, 39–83
- LDAP user lookup caching
 - ldapcachesize, 27–16
 - ldapcachettl, 27–16
- Mapping tables
 - Large, 37–18
- Measuring
 - log_queue_time MTA option, 39–266
- Message Store
 - dbtmpdir Message Store option, 16–11, 16–11
- MTA options
 - File handling, 39–171
 - LDAP and URL lookup caching, 39–152
- SNMP
 - directoryscan SNMP option, 60–2
- TCP/IP channels
 - MAX_CLIENT_THREADS TCP/IP-channel-specific option, 49–30
- Tuning, 56–1
 - buffer_size MTA option, 39–171
 - chunk_cache_limit MTA option, 39–177
 - CPU and resources, 56–1
 - dequeue_map MTA option, 39–172
 - describe_cache_limit MTA option, 39–177
 - Disks and files, 56–3
 - Dispatcher, 56–3
 - Job Controller, 56–4
- UFS
 - osync MTA option, 39–174
- ZFS
 - osync MTA option, 39–174

perms pipe option, 52–15

persistent notifytarget option, 24–4

Personal Addressbook (PAB) lookups

- See PAB, 59–1

personalinc channel option, 33–41, 33–76

personalmap channel option, 33–41, 33–76

- use_personal_names MTA option, 39–201

personalomit channel option, 33–41, 33–76

personalstrip channel option, 33–41, 33–76

personal_conversion_size MTA option, 39–180

perusersynclevel Message Store option, 16–7

pin Message Store option, 16–7

Pipe channels, 52–12

- Addressees and their handling, 52–14
- Aliases in LDAP, 52–14
- Command exit codes, 52–13
- Configuration, 52–13

- Channel definition and rewrite rules, 52–13
- Entry match order, 52–17
- LDAP attributes, 52–14
- Options
 - SHELL_TIMEOUT, 52–17
 - SHELL_TMPDIR, 52–17
 - UNIX_STYLE, 52–17
- Permanent delivery failure
 - Command exit codes, 52–13
- Pipe database, 52–15
- pipeuser option in restricted.cnf, 33–62, 33–107
- Profile database, 52–15
- single channel option, 52–13, 52–16
- site-programs directory, 40–4
- Successful delivery
 - Command exit codes, 52–13
- Temporary delivery failure
 - Command exit codes, 52–13
- user channel option, 52–14
- Pipe options, 52–14
 - command, 52–15
 - params, 52–15
 - perms, 52–15
- pipeuser option in restricted.cnf, 33–62, 33–107
- plaintextconvspace MSHTTP option, 29–7
- plaintextmncipher IMAP option, 21–12, 27–19
- plaintextmncipher MSHTTP option, 27–19, 29–12
- plaintextmncipher MTA option, 27–19, 39–217
- plaintextmncipher option, 27–19
- plaintextmncipher POP option, 22–3, 27–19
- plaintextntabsize MSHTTP option, 29–7
- platformhpux smime option, 30–3
- platformlinuxx86 smime option, 30–3
- platformmac smime option, 30–3
- platformsolarissparc smime option, 30–3
- platformwin smime option, 30–3
- plugins option, 2–1
- PMDF API
 - PMDFgetChannelCounters64, 5–51
- PMDF_CHANNEL environment variable
 - test_smtp_master and test_smtp_slave use of, 52–9
- pmxbl options
 - debug, 7–21
- polldelay IMAP option, 21–4, 27–19
- polldelay MMP option, 21–4, 27–19
- pool channel option, 33–106
 - ims-ms channels, 51–1
 - Job Controller operation, 42–2
 - job_pool Job Controller option, 42–17
- POP
 - Autorestart
 - autorestart.enable option, 7–23

- Commands
 - STARTTLS, sslusessl option, 22–4
 - USER/PASS, broken_client_login_charset auth option, 12–2
- Connection thread hold delay time, 7–13
- Disconnect
 - Forcing via maxprotocolerrors POP option, 22–3
- DNS reverse lookup
 - dnsresolveclient base option, 7–16
- External message collection
 - allowcollect MSHTTP option, 29–5
 - maxcollectmsglen MSHTTP option, 29–5
 - popbindaddr MSHTTP option, 29–5
- Logging
 - logprotocolerrors POP option, 22–3
- Options, 22–1
 - actionattributes, 21–11, 23–1
 - actions, 21–11, 23–1
 - allowanonymouslogin, 22–2
 - banner, 22–2
 - bgdecay, 7–4, 27–8
 - bgexcluded, 7–4, 27–8
 - bglinear, 7–4, 27–8
 - bgmax, 7–4, 27–7
 - bgpenalty, 7–4, 27–8
 - connlimits, 27–10
 - defaultdomain, 27–12
 - domainallowed, 22–2, 27–13
 - domainnotallowed, 22–2, 27–14
 - enable, 22–1
 - enableslport, 22–2
 - forcetelemetry, 22–3
 - idletimeout, 22–3
 - lockmailbox, 22–2
 - logprotocolerrors, 22–3
 - logunauthsession, 22–4
 - maxprotocolerrors, 22–3
 - maxsessions, 22–3
 - maxthreads, 22–3
 - numprocesses, 22–3
 - plaintextmncipher, 22–3, 27–19
 - poplogmbxstat, 22–2
 - popstatuskludge, 22–2
 - port, 22–1
 - ssl cachesize, 22–3
 - ssl nicknames, 22–4, 27–27
 - sslport, 22–4
 - sslusessl, 22–4
- Read vs. unread messages
 - popstatuskludge POP option, 22–2
- Server
 - msprobe probe of, 10–2

- SSL
 - enableslport POP option, 22-2
 - sslport POP option, 22-4
- Startup, 22-1
- XQUERYAUTH
 - authservice option, 27-6
- POP before SMTP
 - \$P input flag in AUTH_REWRITE mapping table, 33-33, 33-64, 33-148
 - proxyused switch of test -rewrite utility, 58-37
 - authservice POP proxy/Virtual Domain option, 27-6
 - authservicettl POP proxy/Virtual Domain option, 27-6
 - Domain catchall mapping, 39-149
 - P modifier in MTA message transaction log entries, 55-4
 - popbeforesmtpkludgechannel option, 27-20
 - preauth MMP/IMAP Proxy/POP Proxy/vdomain option, 27-20
 - Testing for use in address access mapping tables, 44-12
- POP Proxy
 - Options, 27-3
 - authcachettl, 27-5
 - authenticationldapattributes, 27-6
 - authenticationserver, 27-6
 - authservice, 27-6
 - authservicettl, 27-6
 - backsideport, 27-6
 - banner, 27-7
 - canonicalvirtualdomaindelimiter, 27-9
 - connecttimeout, 27-10
 - connlimits, 27-10
 - crams, 27-11
 - debugkeys, 27-11
 - defaultdomain, 27-12
 - domainallowed, 27-13
 - domainnotallowed, 27-13
 - domainsearchformat, 27-14
 - hosteddomains, 27-15
 - ldapcachesize, 27-16
 - ldapcachettl, 27-16
 - ldappendingoplimit, 27-16
 - ldaprefreshinterval, 27-17
 - ldaptimeout, DEPRECATED, 27-17
 - ldapurl, DEPRECATED, 27-17
 - loglevel, 27-18, 39-72
 - mailhostattrs, 27-18
 - maxconcurrentconnectionattempts, 27-18
 - preauth, 27-20
 - preauthtimeout, 27-20
 - replayformat, 27-20
 - replaypass, 27-21
 - requireauthenticationserver, 27-21
 - restrictplainpasswords, 27-21
 - searchformat, 27-21
 - spooftemptl, 27-23
 - spooftempfail, 27-23
 - spooftempfail, 27-23
 - ssladjustciphersuites, 7-11, 27-24
 - sslbacksideport, 27-25
 - sslcachedir, 7-3, 27-25
 - storeadmin, 27-27
 - storeadminpass, 27-27
 - synldap, 27-27
 - tcpaccess, 27-28
 - tcpaccessattr, 27-28
 - timeout, 27-28
 - usergroupdn, DEPRECATED; see ugldapbasedn instead, 27-29
 - use_nslog, 27-29
 - virtualdomaindelimiter, 27-29
 - virtualdomainfile, DELETED; see vdomain options instead, 27-29
 - popbeforesmtpkludgechannel SUBMIT Proxy/vdomain option, 27-20
 - popbindaddr MSHTTP option, 29-5
 - poplogmbxstat POP option, 22-2
 - popstatuskludge POP option, 22-2
- Port
 - 1038
 - Default for MTQP server, 39-212
 - 1080
 - socksport channel option's default, 33-142
 - 1083
 - Default for SpamAssassin SOCKS_PORT option, 45-7
 - 110
 - pop.port default, 22-1
 - 11211
 - Default for memcache: URLs, 39-230
 - Default for memcache: URLs, Sieve duplicate test, 5-24
 - Default for memcache_port MTA option, 39-202
 - 143
 - base.proxyimapport default, 7-16
 - imap.port default, 21-3
 - 161
 - snmp.port default, 60-1
 - 25
 - http.smtpport default, 29-11
 - SMTP client connects to by default, 33-142
 - SMTP server, 41-9
 - 27442

- Job Controller internal communications, 41–10, 42–15
- 389
 - Default for base.ugldapport, 7–14
- 443
 - uwcsslport MSHTTP option, 29–9
- 4570
 - deploymap.port default, 14–1
- 49994
 - watcher.port default, 9–1
- 55000
 - store.dbreplicate.port default, 16–16
- 587
 - SMTP SUBMIT, 49–7
 - SMTP_SUBMIT server, 41–9
- 636
 - Effect on base.ugldapport, 7–14
- 63837
 - Default for MeterMaid server, 46–5
 - metermaid.port default, 46–5
- 783
 - Default for SpamAssassin PORT option, 45–7
- 7997
 - ens.port default, 61–1
- 80
 - retrieveport Message Store archive option, 16–15
 - uwcport MSHTTP option, 29–9
- 8070
 - indexer.port default, 25–1
 - ISS listens by default, 25–1
- 8080
 - da_port default, 29–8
- 8990
 - http.port default, 29–4
- 8991
 - http.sslport default, 29–13
- 993
 - imap.sslport default, 21–12
- 995
 - pop.sslport default, 22–4
- backsideport IMAP Proxy and POP Proxy option, 27–6
- cert_port MSHTTP option, 29–7
- Client
 - Passing to Milster, 45–13
- ENS+SSL
 - sslport ENS option, 61–1
- http.sieve.port option, 29–23
- IMAP+SSL
 - sslport IMAP option, 21–12
- imapport Proxy option, 26–1
- JMQ broker

- jmport notifytarget option, 24–2
- LDAP
 - ugldapport, 7–14
- memcache_port MTA option, 39–202
- PAB
 - ldapport PAB option, 59–2
 - ldap_pab_port MTA option, 39–183
- POP+SSL
 - sslport POP option, 22–4
- proxyimapport base option, 7–16
- proxyport MSHTTP option, 29–7
- server_port MeterMaid client option, 46–5
- server_port SMS smpp_relay option, 53–10
- server_port SMS smpp_server option, 53–13
- SHTTP+SSL
 - sslport MSHTTP option, 29–13
- SMTP
 - noticeport alarm option, 11–1
 - smtpport MSHTTP option, 29–11
 - sslbacksideport IMAP Proxy and POP Proxy option, 27–25
 - tcp_listen block
 - MMP, 27–5
 - tcp_ports Dispatcher option, 41–9
 - tcp_ports Job Controller option, 41–10, 42–15
 - tcp_ports option, 41–9
 - tcp_ports SMS smpp_relay option, 41–9, 53–10
 - tcp_ports SMS smpp_server option, 41–10, 53–13
- port channel option, 33–139, 33–142
 - AUTH_ACCESS mapping table \$P flag, 49–38
- port Deployment Map option, 14–1
- port ENS option, 61–1
 - Match ensport notifytarget option, 24–2
- port IMAP option, 21–3
- port indexer option, 25–1
- port Job Controller option, 42–13
- port Message Store dbreplicate option, 16–16
- port MSHTTP option, 29–4
- port MSHTTP sieve option, 29–23
- port POP option, 22–1
- port SNMP option, 60–1
- port Watcher option, 9–1
- Ports
 - 1344
 - ICAP server, 32–1
- postdatedmode Message Store archive option, 16–15
- PostFix
 - XCLIENT SMTP extension
 - *xclient* channel options, 33–76, 33–132, 33–153
- postheadbody channel option, 33–98
- postheadonly channel option, 33–98

-
- Postmaster
 - Address
 - \$H flag in REVERSE mapping table, 35–53
 - aliaspostmaster channel option, 33–93
 - Case insensitive local-part, 47–26
 - Case-insensitive, 39–55
 - Channel options, 33–93
 - Emitted, 47–26
 - Emitted, Default, 47–26
 - msprobe alarm messages, noticercpt alarm option, 11–1
 - msprobe alarm messages, noticesender alarm option, 11–1
 - Owner of system level Sieves, duplicate test, 5–25
 - Owner of system level Sieves, Sieve duplicate test, 39–230
 - Owner of system level Sieves, SIEVE_EXTLISTS mapping table probes, 5–29
 - Required, 35–9, 47–25
 - returnaddress channel option, 33–97
 - returnpersonal channel option, 33–97
 - return_address MTA option, 39–215
 - return_personal MTA option, 39–216
 - user_case MTA option, 39–64
 - Per-domain
 - Address reversal, 35–50
 - ldap_domain_attr_report_address MTA option, 39–148
 - Warning messages, 47–1
 - post_debug MTA option, 39–75
 - post_off MTA option, 39–185
 - Mailing list members, 36–22
 - preauth option, 27–20
 - preauthtimeout IMAP Proxy/POP Proxy/SUBMIT Proxy option, 27–20
 - preferpoll base option, 7–15, 27–20
 - preferpoll MMP option, 7–15, 27–20
 - prefix_search indexer option, 25–3
 - priority notifytarget option, 24–4
 - probe options, 10–2
 - Process channel, 52–18
 - \$R input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148
 - Notification messages, 47–22
 - Sieve redirect action, 5–40
 - Used for notification messages, 52–18
 - Processing jobs
 - Log file purge job
 - Scheduler task, 8–2
 - Return job
 - *notices channel options, 33–97
 - Expiry-date:, alias_expiry alias option, 35–15
 - Expiry-date:, EXPIRY alias file named parameter, 35–33
 - Expiry-date:, expirysource channel option, 33–68, 33–105
 - MTA transaction log file rollover, 33–85
 - return_units MTA option, 39–216
 - Scheduler task, 8–2
 - processsecuritymultiparts channel option, 33–45
 - process_substitutions MTA option, 39–99
 - Effect on ldap_moderator_url attribute's value, 39–134
 - Profile database, 52–15
 - projectid base option, 7–18
 - projectid MTA option, 39–174
 - Protections
 - Alias database, 35–44
 - Alias file
 - Include files, 35–26
 - General database, 34–25
 - Proxies, 1
 - Proxy options, 26–1
 - httpadminpass
 - DEPRECATED; see proxyadminpass instead, 26–1
 - imapadmin, 26–1
 - imapadminpass, 26–1
 - imapport, 26–1
 - storehostlist, 26–2
 - proxyadmin base option, 7–15
 - Host-specific override by imapadmin option, 26–1
 - proxyadminpass base option, 7–15
 - Host-specific override by imapadminpass option, 26–1
 - proxyimapport base option, 7–16
 - proxyimapssl base option, 7–16
 - proxyserverlist base option, 7–16
 - proxytrustmailhost base option, 7–16
 - PureMessage IP blocker options
 - debug, 7–21
 - Purge (MTA log files) job
 - crontab Scheduler task option, 8–3, 16–24
 - purgemsg notifytarget option, 24–5
 - pwchangeurl base option, 7–8, 7–9
- ## Q
- qm utility
 - Notification message generation, 47–4
 - Queue cache database
 - cache -sync utility, 58–6
 - cache -walk utility, 58–8
 - max_cache_messages Job Controller option, 42–12

- queue_cache_mode MTA option, 39–174
- queuedir msprobe option, 10–1
- queuemax Message Store dbreplicate option, 16–17
- queue_cache_mode_3_files MTA option, 39–174
- Quota
 - Admin bypass
 - adminbypassquota IMAP option, 21–3
 - Bypassing for delivery
 - deliveryflags channel option, 33–108, 33–123
 - capability IMAP proxy option
 - IMAP QUOTA extension, 27–9
 - capability_quota IMAP option
 - IMAP QUOTA extension, 21–8
 - defaultmailboxquota Message Store option, 16–11
 - defaultmessagequota Message Store option, 16–11
 - Folder
 - enable folderquota option, 16–20
 - IMAP QUOTA extension
 - capability IMAP proxy option, 27–9
 - capability_quota IMAP option, 21–8
 - LMTP implementation, 49–14
 - Message type
 - enable typequota option, 16–21
 - MeterMaid connection quota
 - quota local_table option, 46–3
 - quota_time local_table option, 46–3
 - MeterMaid throttle parameter
 - check_memcache.so use, 37–25, 37–27
 - Over quota Message Store notification, 47–3
 - Over quota SMTP rejection
 - error_text_over_quota MTA option, 39–161
 - Over quota status
 - ldap_user_mail_status MTA option, 39–104
 - SMTP rejection, use_permanent_error MTA option, 39–169
 - OverQuota event notification
 - overquota notifytarget option, 24–5
 - overquotastatus Message Store option, 16–7
 - Pitfall of support for deferred message processing, 33–102
 - quotaenforcement Message Store option, 16–12
 - quotaexceededmsginterval Message Store option, 16–12
 - quotagraceperiod Message Store option, 16–12
 - quotanotification Message Store option, 16–12
 - quotawarn Message Store option, 16–13
 - Sieve throttling
 - :quota memcache parameter, 5–11, 5–53
 - :quotatimeout memcache parameter, 5–5, 5–5, 5–11, 5–11, 5–11, 5–53
 - UnderQuota event notification

- underquota notifytarget option, 24–5
- User
 - Disk quota, mailQuota LDAP attribute, 39–106, 39–124
 - folderquota Message Store options, 16–20
 - Mailbox quota, defaultmailboxquota Message Store option, 16–11
 - Maximum messages per folder, maxmessages Message Store option, 16–7
 - Maximum number of folders, maxfolders Message Store option, 16–6
 - Message quota, defaultmessagequota Message Store option, 16–11
 - messagetype Message Store options, 16–20
 - Notification of overquota, IMAP ALERT, 47–3
 - Over quota notification, Message Store generation of, 47–3
 - Over quota notification, quotaexceededmsg Message Store option, 16–13
 - Over quota status, Customizing MTA error text, 39–157
 - Over quota status, error_text_over_quota MTA option, 39–161
 - Over quota status, IMAP ALERT message, 47–3
 - Over quota status, inetUserStatus LDAP attribute, 39–104
 - Over quota status, mailUserStatus LDAP attribute, 39–104
 - Over quota status, quotaoverdraft Message Store option, 16–8
 - Over quota status, Reported to entire group membership, 36–17
 - Over quota status, SMTP rejection and overquotastatus Message Store option, 16–7
 - Over quota status, SMTP rejection text, 39–161
 - Over quota status, use_permanent_error MTA option, 39–169
 - Per message type, 16–22
 - Per message type, Example, 16–21
 - Per-message size quota, mailMsgQuota LDAP attribute, 39–125
 - quotaroot Message Store messagetype mtindex option, 16–22
 - subdirs channel option on ims-ms channel, 51–1
 - typequota Message Store options, 16–20
- Warning message
 - Generated by Message Store and directly deposited, 47–2
 - Message Store notification, 47–3
- quota local_table MeterMaid option, 46–3

quotaenforcement Message Store option, 16–12
quotaexceededmsg Message Store option, 16–13
quotaexceededmsginterval Message Store option, 16–12
quotagraceperiod Message Store option, 16–12
quotanotification Message Store option, 16–12
quotaoverdraft Message Store option, 16–8
 Notification that a Message Store user is overquota, 47–3
quotaroot Message Store message type mtindex option, 16–22
quotawarn Message Store option, 16–13
 Notification that a Message Store user is overquota, 47–3
quota_time local_table MeterMaid option, 46–3

R

Random number generation

 Mapping tables templates, 37–10
 strongrandom Sieve action, 5–18

randommx channel option, 33–137

rbac base option, 7–17

Read receipts

 read_receipt_off MTA option, 39–272
 read_receipt_on MTA option, 39–272

readmsg notifytarget option, 24–5

readreceiptmail channel option, 33–98

readsigncert smime option, 30–6

read_receipt_off MTA option, 39–272

read_receipt_on MTA option, 39–272

rebuild_parallel_channels Job Controller option, 42–13

Receipt requests

 read_receipt_off MTA option, 39–272
 read_receipt_on MTA option, 39–272

receivedfor channel option, 33–74

receivedfrom channel option, 33–74

receivedstate channel option, 33–77

 Conversion channel, 38–5

 filter_discard channel, 52–7

received_domain MTA option

 Limiting emission of internal host names, 57–2

 Local channel official_host_name, 52–2

Recipe language, 4–1

 Special symbolic names, 3–1

Recipient access mapping tables, 44–7

recipientcutoff

 Effect set via address access mapping tables, 44–9

recipientcutoff channel option, 33–87, 33–120

recipientlimit

 Effect set via address access mapping tables, 44–9

recipientlimit channel option, 33–87, 33–120

 error_text_recipient_over MTA option, 39–161

record_lifetime sms_gateway option, 53–4

refuseehlo channel option, 33–118

refusenotary channel option, 33–96, 33–131

registerindices SNMP option, 60–3

rejectsmtpplonglines channel option, 33–133

 error_text_smtp_lines_too_long MTA option, 39–166

relaxheadertermination channel option, 33–73

Relay blocking

 See SMTP relay blocking, 49–48

relay channel option, 33–119

Relaying to an outbound gateway host

 See daemon channel option, 33–61, 33–136

remotehost channel option, 33–36, 33–67

replayformat MSHTTP option, 27–21, 29–4

replaypass mmp/imaproxy/popproxy option, 27–21

reportboth channel option, 33–98

reportdir Message Store archive option, 16–15

reporthead channel option, 33–98

reportnotary channel option, 33–98

reportsuppress channel option, 33–98

Reprocess channel, 52–18

 \$R input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148

 Address *_ACCESS mapping tables

 Probe as prior channel, 52–19

 Debug output, 33–85, 52–19

 domain_failure MTA option and temporary LDAP errors conditions, 34–30

 Force routing via delivery_option, 39–93

 Forced routing when LDAP unavailable

 domain_failure MTA option, 39–79

 ldap_reprocess MTA option, 39–130

 Logging, 52–19

 log_process MTA option, 52–19

 Operation as prior channel, 52–19

 Logging, 52–19

 Retrieving messages from the filter_discard channel, 52–8

 Routing due to presence of mgrpAuthPassword, 39–133

 Sieve redirect action, 5–40

 spamfilterN_received MTA options, 39–239

 transactionlimit channel option, 52–19

Require Recipient Valid Since

 See SMTP, Extensions, RRVs, 33–35, 33–118

requireauthenticationserver auth option, 27–21

requireauthenticationserver IMAP Proxy/POP Proxy option, 27–21

resourcetimeout MSHTTP option, 29–10, 29–10

restrictanyone Message Store privatesharedfolders option, 16–25
 restrictdomain Message Store privatesharedfolders option, 16–25
 restricted channel option, 33–40
 -restricted switch of test -rewrite utility, 58–37
 restrictplainpasswords mmp/imaproxy/popproxy/vdomain option, 27–21
 resubmit_time local_table MeterMaid option, 46–2
 retainsecuritymultipart channel option, 33–45
 retrieveport Message Store archive option, 16–15
 retrieveserver Message Store archive option, 16–15
 retrievetimeout Message Store archive option, 16–15
 return utility
 Notification message generation, 47–4
 returnaddress channel option, 33–97
 returnenvelope channel option, 33–99
 DNS verification
 test -rewrite utility, 58–35
 error_text_invalid_return_address MTA option, 39–166
 error_text_mailfromdnsverify MTA option, 39–165
 error_text_unknown_return_address MTA option, 39–166
 return_envelope MTA option, 39–156, 39–216
 returnenvelope MTA option
 DNS verification
 test -rewrite utility, 58–35
 returnpersonal channel option, 33–97
 return_personal MTA option, 39–216
 return_*.txt files, 47–10
 return_address MTA option
 test -rewrite utility's -from switch, 58–30
 return_cleanup_period MTA option, 39–271
 return_debug MTA option, 39–75
 return_delivery_history MTA option, 39–215
 return_envelope MTA option, 39–155, 39–215
 returnenvelope channel option, 33–99
 return_header.opt file, 47–10
 return_job
 crontab Scheduler task option, 8–3, 8–4
 Enable scheduling of
 enable Scheduler task:return_job option, 8–4
 Options, 8–4
 return_option.opt file, 47–13
 RETURN_PERSONAL option
 return_personal MTA option, 39–216
 return_split_period MTA option, 39–270
 return_units MTA option, 39–216
 Interpretation of *notices channel options, 33–97
 return_verify MTA option, 39–75
 reverse channel option, 33–41
 Reverse database, 35–50
 comment_chars MTA option, 39–171
 MTA options
 comment_chars, 39–171
 reverse_data_size, 39–180
 string_pool_size_3, 39–181
 use_text_databases, 39–175
 reverse_database_url MTA option, 39–203
 reverse_data_size MTA option, 39–180
 string_pool_size_3 MTA option, 39–181
 use_reverse_database MTA option, 39–62, 39–200
 use_text_databases MTA option, 39–175
 reverse_address_cache_size MTA option, 39–153
 reverse_address_cache_timeout MTA option, 39–153
 reverse_database_url MTA option, 39–203
 Address reversal, 35–52
 reverse_data_size MTA option, 39–180
 reverse_url MTA option, 39–88
 revocationunknown smime option, 30–6
 rewrite group, 34–2
 Rewrite rules, 34–1
 Callout routines, 37–24
 Direct LDAP domain lookup (\$V)
 domain_match_cache_size MTA option, 39–13, 39–152
 Direct LDAP domain lookups, 34–29
 Domain database
 imta_domain_database MTA option (DELETED), 40–8
 Domain literal handling, 34–8
 Host/domain with trailing dot, 34–5
 LMTP channels, 39–95
 Operation of, 34–2
 Apply rewrite rule template, 34–7
 Extraction of the first host/domain specification, 34–3
 Failure to match, 34–8
 Finishing rewriting process, 34–7
 Scan for a domain match, 34–5
 Patterns, 34–5
 \$!, 34–11
 \$%, 34–11
 \$*, 34–11
 \$, ims-ms channels, 51–3
 *, 34–12
 ., 34–13
 Domain literal match-all, 34–12
 Initial match-all, 34–11
 Match any address, 34–12
 Match bang-style addresses, 34–11

- Match exact numbers of domain components, 34–12
- Match percent hacks, 34–11
- Short form host name, 34–12
- Syntax, 34–9
- Syntax, Case-insensitive matching, 34–9
- Tagged sets of rewrite rules, 34–13
 - [], 34–12
- Percent signs, 34–4
- rewrite group, 34–2
- Selectively analyze and modify usernames, 34–25
- Source channel-specific
 - BSMTP channel example, 50–3
- Syntax checking of resulting address, 34–8
- Template, 34–7
 - Case preserving, 34–14
 - Formats, 34–14
 - Ordinary format, 34–14
 - Repeated rewriting format, 34–15
 - Specified route formats, 34–15
 - Syntax, 34–14
- Template control sequences, 34–16
 - \$,, 34–32
 - \$1M, 34–27
 - \$1M, domain_failure MTA option usage of, 39–80
 - \$1N, 34–27
 - \$1~, 34–31
 - \$1~, domain_failure MTA option usage of, 39–80
 - \$:, 34–32
 - \$;, 34–32
 - \$>, 34–32
 - \$?, 34–34
 - \$/, domain_failure MTA option usage of, 39–80
 - \$A, 34–28
 - \$B, 34–28
 - \$C, 34–27
 - \$E, 34–28
 - \$F, 34–28
 - \$I, 34–32
 - \$M, 34–26
 - \$M, domain_failure MTA option usage of, 39–80
 - \$N, 34–26
 - \$nT, 34–32
 - \$nxxxyyy?, 34–34
 - \$O, 34–32
 - \$P, 34–28
 - \$Q, 34–27
 - \$R, 34–28
 - \$S, 34–28
 - \$T, 34–33
 - \$V, domain_failure MTA option, 39–79
 - \$V, domain_match_cache_size MTA option, 39–13, 39–152
 - \$V, domain_match_url MTA option, 39–80
 - \$X, 34–28
 - Alias-sensitive, 34–32
 - alias_magic override, 34–32
 - Destination channel-specific, 34–27
 - Direction-specific, 34–28
 - Domain found/not-found in LDAP, 34–29
 - Error text, 34–34
 - Host location-specific, 34–28
 - List-name matching, 34–32
 - Location-specific, 34–28
 - Message MT-PRIORITY comparisons, 34–33
 - Message size comparisons, 34–33
 - Source channel-specific, 34–26
 - Tag value, 34–33
 - TLD comparison, 34–32
- Template substitutions, 34–7, 34–16
 - \$!n, 34–20
 - \$#n, 34–20
 - \$\$, 34–21
 - \$%, 34–21
 - \$&n, 34–20
 - \$(...), 34–24
 - \$*n, 34–20
 - \$., 34–24
 - \$.text., 34–24
 - \$0U, 34–19
 - \$1U, 34–19
 - \$<...>, 34–26
 - \$=, 34–22
 - \$@, 34–21
 - \$D, 34–20
 - \$D, \$H, \$U, 34–7
 - \$G, 34–21
 - \$H, 34–20
 - \$L, 34–20
 - \$n<...>, 34–26
 - \$nD, 34–20
 - \$nG, 34–21
 - \$nH, 34–20
 - \$nY, 34–26
 - \$U, 34–19
 - \$W, 34–26
 - \$Y, 34–26
 - \$[...], 34–25
 - \$\, 34–21
 - \$]...[, 34–22
 - \$^, 34–21

- \$_, 34-21
- \$_ turns off LDAP URL encoding, 34-23
- \${...}, 34-25
- Case changing, 34-21
- Cases of temporary LDAP lookup failure, 34-24
- Domain, 34-20
- General database, 34-24
- Hash of argument, 34-26
- Host, 34-20
- IP literal, 34-20
- LDAP query results, 34-22
- LDAP URL encoding, 34-22
- Literal character, 34-21
- Local-part, 34-19
- Mapping table callout, 34-25
- Routine callout, 34-25
- Subaddress, 34-19
- Subdomain single field, 34-20
- Transport information, 34-26
- Unique string, 34-26
- Username, 34-19
- Testing of
 - test -rewrite utility, 58-28
- re_pattern backup_group option, 19-1
- RFC 1891-1894 (NOTARY)
 - content_return_block_limit MTA option, 39-207, 39-213
- rfc2231compliant MSHTTP option, 29-5
- rfc822headerallow8bit base option, 7-15
- Role-Based Access Controls
 - rbac base option, 7-17
- rolename option, 2-1
- rollingdbbackup Message Store option, 16-8
- rollovermanager
 - Options, 15-1
 - enable, 15-1, 15-1
- rolloverpolicy logfile option, 7-20
- rollovertime logfile option, 7-20
- Rose, Marshall
 - Fundamental axiom of management
 - MTA counters, 55-24
- routelocal channel option, 33-42
 - Compared to localbehavior, 33-39
 - Removal of source routes during rewriting, 34-8
- route_to SMS gateway_profile option, 53-8
- route_to_routing_host MTA option
 - Effect on mailRoutingHosts interpretation, 39-144
- Routing
 - \$~ flag in FROM_ACCESS mapping, 44-13, 47-23
 - aliasdetourhost channel option, 33-30, 33-60

- Alternate conversion channel, 38-4
 - aliasdetourhost used in conjunction, 33-30, 33-60
 - Notification messages, 47-23
- Gateway systems, 49-46
 - Example, 49-47
 - vs. Smart host vs. Spam/virus filter box, 49-47
- ldap_detourhost_optin MTA option, 39-123
- See also daemon channel option, 33-61, 33-136
- RRVS
 - See SMTP, Extensions, RRVS, 33-35, 33-118
- rules channel option, 33-41
 - Source channel-specific rewriting, 34-26
- run_as_server Deployment Map option, 14-2

S

- S/MIME, 1
- S/MIME options, 30-1
 - alwaysencrypt, 30-3
 - alwaysign, 30-4
 - appletlogging, 30-7
 - certurl, 30-2
 - checkoverssl, 30-5
 - crl_dir, 30-4
 - crlenable, 30-4
 - crmappingurl, 30-5, 30-5
 - crlurllogindn, 30-4
 - crlurlloginpw, 30-5
 - crlusepastnextupdate, 30-7
 - enable, 30-1
 - loginpw, 30-2
 - platformhpux, 30-3
 - platformlinuxx86, 30-3
 - platformmac, 30-3
 - platformsolarissparc, 30-3
 - platformwin, 30-3
 - readsigncert, 30-6
 - revocationunknown, 30-6
 - sendencryptcert, 30-6
 - sendencryptcertrevoked, 30-6
 - sendsigncert, 30-7
 - sendsigncertrevoked, 30-7
 - sslrootcacertsurl, 30-2, 30-2
 - timestampdelta, 30-5
 - trustedurl, 30-2
 - usercertfilter, 30-1
- safe_tcl_mode MTA option, 39-273
- SASL
 - Mechanisms
 - ANONYMOUS, allowanonymouslogin
 - MSHTTP option, 29-11
 - EXTERNAL, externalidentity channel option, 33-147

EXTERNAL, EXTERNAL_IDENTITY TCP/IP-channel-specific option, 49–21

PLAIN, authpassword and authusername channel options, 33–147

PLAIN, AUTH_PASSWORD and AUTH_USERNAME TCP/IP-channel-specific options, 49–21

saslpassth channel option, 33–154

saslruleset channel option, 33–154

saslswitchchannel channel option, 33–82, 33–154

Effect nullified by XUNAUTHENTICATE SMTP command, 33–83, 33–155

sasltrustauth channel option, 33–154

scan_channel MTA option, 39–170

imexpire, 45–17

scan_originator MTA option, 39–170

imexpire, 45–17

scan_recipient MTA option, 39–170

imexpire, 45–17

Scheduler, 1

Scheduler options, 8–1

enable, 8–1

enablelog, 8–1

task

crontab, 8–2

enable, 8–2

msprobe, enable, 10–1

return_job, crontab, 47–4

return_job, enable, 8–4, 47–4

snapshot, enable, 8–5

snapshotverify, enable, 8–5

task:expire

crontab, 8–3, 16–18

task:msprobe

crontab, 8–3, 10–2

task:purge

crontab, 8–3, 16–24

task:return_job

crontab, 8–3, 8–4

task:snapshot

crontab, 8–4, 8–5

task:snapshotverify

crontab, 8–4, 8–5

Scheduler task options, 8–1

crontab, 8–2

enable, 8–2

Schema

Extending

ldap_domain_known_attributes MTA option, 7–6, 39–83

mailAutoReply* attributes on mailing lists and groups, 39–93

scriptlimit channel option, 33–111

searchfilter auth option, 12–1

Compared with ldap_schematag, 39–90

searchfordomain auth option, 12–1

searchformat mmp/imaproxy/popproxy/vdomain option, 27–21

secondclassafter channel option, 33–100

secondclassblocklimit channel option, 33–114

second_class_block_limit MTA option, 39–210, 39–220

secret base option, 7–15

secret ENS option, 61–2

secret Job Controller option, 42–14

secret MeterMaid option, 46–5

secret Watcher option, 9–1

sectoken options, 13–1

tokenpass, 13–1

Security token options

See sectoken options, 13–1

seen Message Store expire rule option, 16–20

seenckpinterval Message Store option, 16–8

seenckpstart Message Store option, 16–8

select_re SMS gateway_profile option, 53–8

Semicolon

Comment line in MTA configuration files

comment_chars MTA option, 39–171

sendencryptcert smime option, 30–6

sendencryptcertrevoked smime option, 30–6

Sender Permitted From

See SPF lookups, 39–240

Sender Policy Framework

See SPF lookups, 39–240

Sender Rewriting Scheme

See SRS, 39–244

sendetrn channel option, 33–131

sendpost channel option, 33–94, 47–1

sendsigncert smime option, 30–7

sendsigncertrevoked smime option, 30–7

sensitivitycompanyconfidential channel option, 33–107

sensitivitynormal channel option, 33–107

sensitivitypersonal channel option, 33–107

sensitivityprivate channel option, 33–107

separate_connection_log MTA option, 39–270

Server Side Rules database

imta_ssr_database, 40–8

serverdomainalert IMAP Proxy option, 27–22

serversidewastebasket Message Store deleted option, 16–26

servertimeout SNMP option, 60–3

serveruid base option, 7–11

server_host Deployment Map option, 14–1

server_host icapservice option, 32–1

server_host indexer option, 25–2

server_host MeterMaid Client option, 46–4
server_host smpp_relay option, 53–10
server_host smpp_server option, 53–13
server_nickname MeterMaid client remote_table option, 46–5
server_port icapservice option, 32–1
server_port MeterMaid client option, 46–5
server_port MeterMaid remote_server option, 46–5
server_port smpp_server option, 53–13
server_port SMS smpp_relay option, 53–10
server_receive_timeout smpp_relay option, 53–10, 53–10
Service conversions, 38–26
 BSMTP channels, 50–4
 SERVICE-CALL conversion entry parameter, 38–8
 SERVICE-COMMAND conversion entry parameter, 38–8
serviceadmingroupdn Message Store option, 16–13
serviceconversion channel option, 33–55
service_name icapservice option, 32–1
setacl notifytarget option, 24–5
sevenbit channel option, 33–53, 33–125
sharedfolders Message Store option, 16–8
shareflags Message Store privatesharedfolders option, 16–25
Shell utilities, 58–2
showunreadcounts MSHTTP option, 29–6
Sieve filters, 5–1, 5–2, 44–1
 :detail
 subaddress extension, 5–43
 :list match type, 5–29
 :regex match type, 5–18
 :user
 subaddress extension, 5–43
Actions
 addconversiontag, 5–18, 5–48
 addflag, 5–37
 addheader, 5–25
 addheader, max_addheaders MTA option, 39–225
 addprefix, 5–18, 5–49
 addsuffix, 5–18, 5–49
 addtag, 5–18, 5–49
 adjustcounter, 5–18
 capture, 5–18, 5–51
 capture, Timing of capture message generation, 47–4
 debug, 5–18
 deleteheader, 5–25
 duplicate, max_duplicates MTA option, 39–226, 39–231
 ereject, 5–28
 ereject, enable_sieve_ereject MTA option, 39–228
 fileinto, 5–36
 fileinto, :flags, 5–37
 fileinto, max_fileintos MTA option, 39–226
 hold, 5–18, 5–51
 importanceadjust, 5–18, 5–52
 importancetest, 5–18
 jettison, 5–18, 5–22
 keep, :flags, 5–37
 memcache, 5–53
 metermaid, 5–57
 monitor, 5–18
 monitor, Synonym for capture, 5–51
 nonotify, 5–18, 5–40
 notify, 5–39
 notify, Cancelled in user Sieves by ereject, reject, or refuse, 5–29
 notify, jettison cancels, 5–23
 notify, max_notifys MTA option, 39–226
 novation, 5–18, 5–44
 override, 5–18
 redirect, 5–40
 redirect, :copy parameter, 5–41
 redirect, :keepmailfrom parameter, 5–41
 redirect, :list, 5–29
 redirect, :notify, 5–41
 redirect, :resent and :noresent parameters, 5–41
 redirect, :resetmailfrom parameter, 5–41
 redirect, :ret, 5–41
 redirect, Extensions to, 5–41
 redirect, max_redirect_addresses MTA option, 39–226
 redirect, Reprocess channel vs. process channel, 52–18
 refuse, 5–28
 refuse, Sieve hierarchy, 5–67
 reject, Redefined by ereject extension, 5–28
 removeconversiontag, 5–18, 5–48
 removeflag, 5–37
 replaceheader, 5–25
 require, Not needed after ihave, 5–37
 set, 5–46
 set, :encodeurl, 5–40, 5–47
 set, :length, 5–46
 set, :lower, 5–46
 set, :lowerfirst, 5–46
 set, :quoteregex, 5–47
 set, :quotewild, 5–47
 set, :quotewildcard, 5–47
 set, :upper, 5–46
 set, :upperfirst, 5–46

- setconversiontag, 5–18, 5–48
- setdate, 5–46
- setenvelopefrom, 5–18, 5–61
- setenvelopefrom, Syntax, 5–8
- setflag, 5–37
- setmtpriority, 5–18, 5–62
- setnotify, 5–18
- setoperation, 5–18, 5–61
- setpriority, 5–62
- setreturn, 5–18
- spamadjust, 5–18, 5–42
- strongrandom, 5–18
- transactionlog, 5–18
- translate, 5–62
- vacation, autoreply_timeout_default MTA option, 39–65
- vacation, jettison cancels, 5–23
- vacation, Timing of response message generation, 47–4
- virusset, 5–18, 5–42
- warn, 5–63
- addconversiontag action, 5–48
 - Example, 5–49
- addflag action, 5–37
- addprefix action, 5–49
- address test, 5–21
 - :aindex, 5–21
 - :anychild, 5–38
 - :comment, 5–21
 - :detail, 5–21
 - :display, 5–21
 - :index, 5–23
 - :last, 5–23
 - :mime, 5–38
 - :raw, 5–18, 5–21
 - :text, 5–18, 5–21
 - :user, 5–21
- addsufffix action, 5–49
- alias_filter alias option, 35–15
- Assignment statements, 5–46
 - Integer values, 5–
 - List values, 5–
- Authenticated originator
 - Access to, 5–27
- Autoreply external vs. internal
 - Access to, 5–27
- body test, 5–21
 - :regex match type not supported, 5–61
- capture action, 5–51
 - Format of message, 47–12
 - Timing of capture message generation, 47–4
- Channel name
 - Access to, 5–27
- Channel options, 33–109
- Comparators, 5–51
 - Collapsing white space, 5–52
 - i;ascii-casemap, 5–51
 - i;ascii-casemap-collapse, 5–52
 - i;ascii-integer, 5–52
 - i;ascii-numeric, 5–51
 - i;ascii-numeric, Example scriptlet, 36–6, 45–17
 - i;octet, 5–51
 - i;octet-collapse, 5–52
 - utf-8, :regex match type not supported, 5–61
- Control structures
 - error, 5–37
 - foreverypart, 5–38
 - loop, 5–52
 - require, strict_require MTA option, 39–229
- Conversion tags, 5–48
 - Example, 5–49
- copy extension, 5–22
- Counters, 5–50
- currentdate test
 - :list, 5–24
 - :zone, 5–24
 - Example, 5–34
- date extension, 5–23
 - Example, 5–34
- date test
 - :list, 5–24
 - :originalzone, 5–24
 - :zone, 5–24
- debug action
 - mm_debug MTA option, 5–18
- Debugging, 39–231
 - debug action, 5–18
 - filter_debug MTA option, 39–74, 39–231
- decode_encoded_words MTA option, 39–224
- deleteheader action
 - Limiting emission of internal host names, 57–4
- discard action
 - deliveryflags channel option, 33–108, 33–123
 - Disabled on messages retrieved from filter_discard channel, 52–8
 - filter_discard MTA option, 39–224
 - Force via address access mapping tables, 44–9
- Domain
 - ldap_domain_attr_filter MTA option, 39–148
- DSN parameters
 - redirect action, :notify and :ret parameters, 5–41
- duplicate test
 - duplicate_maximum_timeout MTA option, 39–230

- duplicate_minimum_timeout MTA option, 39–230
- duplicate_tracking_url MTA option, 39–230
- Issues, log_filter MTA option, 5–25
- memcache update timing, 5–25
- editheader extension
 - Alternative to header trimming, 33–155
 - Compared to use of mgrpAddHeader group LDAP attribute, 39–139
 - Compared to use of mgrpRemoveHeader group LDAP attribute, 39–139
- envelope test, 5–26
 - conversiontag, Example, 5–49
 - from, Example, 5–33
 - Head-of-household use, 5–73
- environment extension
 - Custom items, 5–15
 - domain, 5–15
 - host, 5–15
 - location, 5–15
 - name, 5–15
 - phase, 5–15
 - remote-host, 5–15
 - remote-ip, 5–15
 - version, 5–15
 - vnd.oracle.last-verdict, 5–15
 - vnd.oracle.message-hash, 5–15
 - vnd.oracle.mt-priority, 5–15
 - vnd.oracle.operation-type, 5–15
 - vnd.oracle.reevaluate, 5–15
 - vnd.oracle.tracking-id, 5–15
 - vnd.sun.authenticated-sender-address, 5–15
 - vnd.sun.authenticated-sender-id, 5–15
 - vnd.sun.autoreply-internal, 5–15
 - vnd.sun.destination-channel, 5–15
 - vnd.sun.source-channel, 5–15, 5–15
- Environment extension, 5–27
 - Custom items via \$+E *_ACCESS flag, 5–28
 - domain, 5–27
 - domain, ldap_default_domain MTA option, 39–83, 39–97
 - domain, received_domain MTA option, 39–83, 39–97, 39–222
 - host, 5–27
 - location, 5–27
 - name, 5–27
 - phase, 5–27
 - remote-host, 5–27
 - remote-ip, 5–27
 - version, 5–27
 - vnd.oracle.last-verdict, 5–27
 - vnd.oracle.mt-priority, 5–27
 - vnd.oracle.operation-type, 5–28
 - vnd.sun.authenticated-sender-address, 5–27
 - vnd.sun.authenticated-sender-id, 5–27
 - vnd.sun.autoreply-internal, 5–27
 - vnd.sun.destination-channel, 5–27
 - vnd.sun.source-channel, 5–27
- error control structure, 5–37
- Errors
 - return_error.txt file, 47–13
- Example scriptlets
 - := assignment, 5–63
 - :comparator "i;ascii-numeric", 36–6, 45–17
 - :flags, 5–37
 - addprefix action, 5–26
 - addprefix extension, 5–39
 - adjustcounter action, 5–51
 - Attachment rejection, 36–6
 - Automatic response to list postings, 36–8
 - Conditional capture, 5–35
 - Conversion tag manipulation, 5–49
 - Counters, 5–51
 - currentdate, 5–23, 5–34
 - Custom (mapping-defined) heloname environment item, 5–28
 - Date-based redirects, 5–34
 - DMARC broken usage and corresponding address adjustment, 5–26
 - envelope, 36–6
 - envelope "to", 5–43
 - Envelope test on conversion tags, 5–49
 - Envelope test on subaddress, 36–5
 - Envelope test whether originator in PAB, 5–33
 - envelope, Moderated mailing list, 36–5
 - Environment test on custom (mapping-defined) item, 5–28
 - environment test, host, 5–51
 - Expression in place of simple argument, 5–63
 - External list, 5–33, 5–33, 5–35
 - fileinto, 5–43
 - foreverypart extension, 5–39
 - header test, Authentication-results, 5–51
 - Holiday redirection, 5–34
 - imap4flags, 5–37
 - Logging header line, 39–232, 39–269
 - loop construct, 5–52
 - Manifest, 5–39
 - Mapping table test, 5–64
 - memcache, 5–54
 - mime :anychild, 36–6
 - mime extension, 5–39
 - Moderate medium-sized messages; reject large messages, 36–7
 - Moderate possible spam; reject likely spam, 36–6

- Multi-line string using the text: keyword, 5-26
- notify action, 5-40
- notify, Moderated mailing list, 36-7
- PAB lookups, 5-33
- PAB white-listing of sender, 5-33
- redirect :copy, 5-22
- redirect :resetmailfrom, 36-6
- redirect :resetmailfrom, Moderated mailing list, 36-5
- Reject messages with non-text parts, 36-6
- reject, Moderated mailing list, 36-5
- reject, QUARANTINE_ACTION Milter option, 45-5
- Rejecting replies, 36-5
- relational, 5-33, 5-42
- Relational extension, Conversion tag count, 5-49
- relational, Date operations, 5-23
- relational, imexpire use, 45-17
- relational, Moderated mailing list, 36-6
- replaceheader action, 5-26
- Replies not allowed, 36-5
- setnotify, 5-61
- setreturn, 5-61
- size, Moderated mailing list, 36-7
- spamadjust, 5-33, 5-33
- spamtest, 5-33, 5-33, 5-42
- spamtest, imexpire use, 45-17
- spamtest, Moderated mailing list, 36-6
- subaddress, 5-43
- subaddress, Moderated mailing list, 36-5
- Test of client HELO/EHLO name, 5-28
- text: keyword and a multi-line string, 5-26
- transactionlog, 39-232, 39-269
- translate, 5-63, 5-63
- vacation, Date checks, 5-23
- vacation, Moderated mailing list, 36-8
- variables, 5-51, 5-64
- Variables substitutions in body test not permitted, 5-21
- variables, Conversion tags, 5-49
- variables, Header line logging with transactionlog, 39-232, 39-269
- variables, replaceheader action, 5-26
- virustest, 5-42
- White-listing known correspondents (PAB addresses), 5-33
- exists test
 - :anychild, 5-38
 - :mime, 5-38
- exitif, 5-52
- Extensions, 5-16
 - :raw and :text modifiers for address and header tests, 5-18
 - Add prefix or suffix text to first text part, 5-18, 5-49
 - addtag, 5-18, 5-49
 - adjustcounter, 5-50
 - Body, :matches match type not supported, 5-21
 - Body, :regex match type not supported, 5-21
 - body, :regex match type not supported, 5-61
 - body, enable_sieve_body MTA option, 39-228
 - body, imexpire, 5-22
 - Body, Restrictions, 5-21
 - Body, Variable substitution not supported in body test arguments, 5-21
 - capture, 5-18, 5-51
 - comparator-*, 5-51
 - Conversion tag operations, 5-18, 5-48
 - copy, 5-22
 - copy, redirect, 5-41
 - Counters value adjustment, 5-18
 - Custom tests defined via mapping tables, 5-63
 - Custom tests via mapping tables, 5-18
 - date, 5-23
 - Date and index, Example, 5-34
 - debug, 5-18
 - duplicate, 5-18, 5-24
 - duplicate, Error logging, 5-63
 - duplicate, MTA options, 39-229
 - editheader, 5-25
 - editheader, Alternative to header trimming, 33-155
 - enotify, 5-39
 - enotify, notify_method_capability test, 5-40
 - enotify, valid_notify_method test, 5-40
 - envelope, 5-26
 - envelope, conversiontag, 5-27
 - envelope-auth, 5-26
 - envelope-dsn, 5-26
 - Environment, 5-27
 - environment, vnd.oracle.last-verdict value of redirect, 5-41
 - ereject, 5-28
 - extlists, 5-29
 - extlists, max_redirect_addresses MTA option, 39-226
 - extlists, redirect action, 5-41
 - extracttext, 5-38
 - foreverypart, 5-38
 - foreverypart, break, 5-38
 - foreverypart, continue, 5-38
 - hold, 5-18, 5-51

- ihave, 5–37, 5–70
- imap4flags, 5–37
- imap4flags, :regex match type not supported with hasflag, 5–61
- Importance operations, 5–18
- importanceadjust, 5–52
- importancetest, 5–52
- index, 5–23
- jettison, 5–18, 5–22
- loop construct, Compared to foreverypart, 5–38
- Loop structure, 5–18
- memcache, 5–53
- memcache, enable_sieve_memcache MTA option, 39–228
- Message capture, 5–18
- metermaid, 5–57
- metermaid, enable_sieve_metermaid MTA option, 39–229
- mime, 5–38
- monitor, 5–18
- nonotify, 5–18, 5–40
- notify, 5–39
- novacation, 5–18, 5–44
- override, 5–18
- redirect-dsn, 5–41
- refuse, 5–28
- refuse, Sieve hierarchy, 5–67
- regex, 5–18, 5–60
- regex, Variables, 5–61
- reject, 5–28
- Relational, 5–41
- setenvelopefrom, 5–18, 5–61
- setenvelopefrom, Syntax, 5–8
- setmtpriority, 5–18, 5–62
- setnotify, 5–18, 5–61
- setoperation, 5–18, 5–61
- setpriority, 5–62
- setreturn, 5–18, 5–61
- spamadjust, 5–18, 5–42
- spamtest, 5–42
- spamtestplus, 5–42
- strongrandom, 5–18
- subaddress, 5–43
- Subaddress, address test, 5–21
- transactionlog, 5–18
- translate, 5–62
- Use of require, strict_require MTA option, 39–229
- vacation, 5–43
- vacation, Error logging, 5–63
- vacation, Why a vacation message was not generated, 5–45
- vacation-seconds, 5–43
- Variables, 5–46
- variables, :regex match type, 5–61
- Variables, Properties of external lists, 5–30, 5–47
- Variables, Substitutions not allowed in body arguments, 5–46
- virusset, 5–18, 5–42
- virustest, 5–42
- warn, 5–63
- External lists, 5–29
 - Example, max_redirect_addresses MTA option, 5–32
 - ldap_ext_* MTA options, 39–182
 - Properties, 5–34
 - spamtest and virustest tests, 5–42
 - Testing, 5–36
- File access error for user Sieve file
 - error_text_sieve_access MTA option, 39–161
- fileinto action, 5–36
 - :copy, 5–22, 5–37
 - :flags, 5–37, 5–37
 - ims-ms channel, 51–2
- fileinto channel option, 33–110
- fileinto vs. redirect, 5–37
- filter_discard channel, 52–7
- filter_discard MTA option, 52–7
- filter_jettison MTA option, 52–7
- Force via address access mapping tables, 44–9
- foreverypart extension, 5–38
- Forwarding via mailForwardingAddress
 - sieve_user_carryover MTA option, 39–101, 39–225
- hasflag test, 5–37
 - :regex match type not supported, 5–61
- Head-of-household, 5–73
 - fileinto, :owner, 5–37
 - ldap_filter_reference MTA option, 39–129
 - ldap_hoh_filter MTA option, 39–97, 39–141
 - ldap_hoh_owner MTA option, 39–97, 39–141
 - ldap_parental_controls MTA option, 39–129
- header test
 - :anychild, 5–38
 - :index, 5–23
 - :last, 5–23
 - :mime, 5–38
 - :raw, 5–18
 - :text, 5–18
 - defer_header_addition MTA option, 39–224
- Hierarchy, 5–65
 - Evaluation, 5–67
 - override action, 5–18
 - Semantics, 5–67

- Types of Sieve scripts, 5–65
- Verdicts, Access to, 5–27
- hold action, 5–51
 - Diagnosing .HELD files, 52–11
- ihave test, 5–37
 - max_notifys MTA option, 39–226
- Implementation internals, 5–72
 - mm_check_function MTA routine, 5–73
 - mm_eval_function MTA routine, 5–73
 - systemfilter, Compiled configuration, 5–72
- index extension, 5–23
- Integers
 - Signed, i;ascii-integer comparator, 5–52
- jettison action, 5–22
 - deliveryflags channel option, 33–108, 33–123
 - Disabled on messages retrieved from filter_discard channel, 52–8
 - filter_jettison MTA option, 39–224
 - Force via address access mapping tables, 44–9
- keep action
 - :flags, 5–37
- Language elements, 5–2, 5–3
- ldap_filter MTA option, 39–129
- ldap_filter_reference MTA option, 39–129
- Logging effect in MTA transaction log
 - log_filter MTA option, 39–231, 39–256
- loop construct, 5–52
- loop structure, 5–18
- Mailing lists
 - FILTER alias file named parameter, 35–33
 - ldap_filter MTA option, 39–129
 - mailSieveRuleSource LDAP attribute, 39–129
 - Owner of, mgrpErrorsTo LDAP attribute, 39–137
- Mapping table tests, 5–63
- Match types
 - :aindex, 5–15, 5–24
 - :contains, 5–15
 - :count, 5–15, 5–17
 - :index, 5–15
 - :is, 5–16
 - :last, 5–15
 - :list, 5–16, 5–29
 - :matches, 5–16
 - :matches, Not supported in body tests, 5–21
 - :regex, 5–16, 5–18
 - :regex, Not supported in body tests, 5–21, 5–61
 - :value, 5–16, 5–17
 - regex, 5–60
 - regex, Variables, 5–61
- Message Store expire rules
 - expiresieve Message Store expire option, 16–13
- Mime extension
 - :contenttype, 5–16
 - :param, 5–16
 - :subtype, 5–16
 - :type, 5–16
- monitor action
 - Synonym for capture, 5–51
- MSHTTP
 - SSL, sslport MSHTTP sieve option, 29–23
- MT-PRIORITY
 - Access to, 5–27
- MTA options, 39–223
 - Caching of Sieves, 39–227
 - decode_encoded_words, 39–224
 - Duplicate recent messages, 39–229
 - Error text, 39–231
 - Interpretation of Sieves, 39–223
 - Language extensions, 39–228
 - Logging and debugging, 39–231
 - See also External filtering context MTA options, 39–170
 - Size limits, 39–225
- nonotify action, 5–40
- notify action, 5–39
 - Invalid recipient addresses, notify_ignore_errors MTA option, 39–225
 - notify_ignore_errors MTA option, 39–225
 - notify_maximum_timeout MTA option, 39–65
 - notify_minimum_timeout MTA option, 39–66
 - Suppressed by nonotify, 5–40
- notify_method_capability test, 5–40
- novacation action, 5–44
 - Disables sending back a vacation message, 5–45
- operation-type
 - Access to, 5–28
- Operators
 - memcache, 5–53
 - memcache, :add, 5–54
 - memcache, :adjustdown, 5–54
 - memcache, :adjustup, 5–55
 - memcache, :append, 5–55
 - memcache, :fetch, 5–56
 - memcache, :prepend, 5–56
 - memcache, :remove, 5–56
 - memcache, :replace, 5–56
 - memcache, :store, 5–57
 - memcache, :throttle, 5–57
 - metermaid, 5–57
 - metermaid, :adjustdown, 5–58
 - metermaid, :adjustup, 5–58, 5–59

- metermaid, :fetch, 5–59
- metermaid, :remove, 5–60
- metermaid, :store, 5–60
- metermaid, :throttle, 5–60
- override action, 5–18, 5–67
- Owner
 - Envelope From address in "capture :message" copy, 54–11
- Owner of
 - :owner tag, 5–37
 - duplicate test, 5–25
 - ldap_hoh_owner MTA option, 5–73, 39–97, 39–142
 - mgrpErrorsTo LDAP attribute on mailing lists, 39–137
 - Sieve syntax error notifications, 47–2
 - SIEVE_EXTLISTS mapping probes, 5–29
- Performance
 - Recipient-specific, vnd.oracle.last-verdict, 5–27
- Processing error report
 - return_error.txt file, 47–13
- Processing priority
 - Access to, vnd.oracle.mt-priority Sieve filter environment item, 5–27
- Random number generation
 - strongrandom, 5–18
- redirect action
 - :copy, 5–22
 - :list, 5–29
 - :list syntax, 5–29
 - defer_header_addition MTA option, 39–224
 - max_redirects, 39–226
 - SIEVE_EXTLISTS mapping probes, 5–30
 - sieve_redirect_add_resent MTA option, 39–225
- Reevaluation
 - vnd.oracle.reevaluate environment item, 5–15
- regex extension
 - Variables, 5–61
- reject action
 - Not recorded in log_filter field of MTA transaction log file, 39–232, 39–257
- relational extension
 - imexpire use, 45–17
- removeconversiontag action, 5–48
 - Example, 5–49
- removeflag action, 5–37
- replaceheader action
 - Limiting emission of internal host names, 57–4
- require action
 - enotify vs. notify, 5–40
 - Example, 5–33
 - Not needed after ihave, 5–37
 - strict_require MTA option, 39–229
- scriptlimit channel option, 33–111
- Server Side Rules (SSR) database storage
 - imta_ssr_database MTA option (DELETED), 40–8
 - ssr: URL type, 1–4
 - ssr_database_url, 39–203
- set action, 5–46
 - :encodeurl, 5–40, 5–47
 - :length, 5–46
 - :lower, 5–46
 - :lowerfirst, 5–46
 - :quoteregex, 5–47
 - :quotewild, 5–47
 - :quotewildcard, 5–47
 - :upper, 5–46
 - :upperfirst, 5–46
- setconversiontag action, 5–48
 - Example, 5–49
- setdate action, 5–46
- setflag action, 5–37
- setmtpriority action
 - Job Controller delivery execution window, 42–16
 - Override MT-PRIORITY value, 39–219
- setpriority action
 - Job Controller delivery execution window, 42–6, 42–16
- sieve_user_carryover MTA option, 39–101, 39–225
- size test
 - Operating on part vs. message, 5–38
 - Syntax of, 5–12
 - Units for value, 39–206
- SMTP AUTH
 - Access to, 5–27
- spamadjust action, 5–42
 - \$_ACCESS flag, 5–42
- spamttest extension
 - imexpire use, 45–17
- spamttest test, 5–42
 - :list, 5–42
 - :percent, 5–42
- string test, 5–46
- Strings
 - text: syntax for entering long, multi-line strings, addprefix and addsuffix actions, 5–49
- subaddress extension, 5–43
 - address test, 5–21
- Subroutines, 5–18, 5–64
- Syntax, 5–3

- Lists, max_sieve_list_size, 39–226
- strict_require MTA option, 39–229
- Strings, max_sieve_string_size, 39–227
- Syntax errors
 - error_text_sieve_syntax MTA option, 39–161
 - Report message, 47–2
 - Report message, Channel filters, 47–2
 - Report message, Head-of-household, 47–2
 - Report message, System filter, 47–2
 - Report message, Timing of generation of, 47–4
 - Report message, User Sieve filters, 47–2
- System-level
 - destinationfilter channel option, 33–109
 - Sieve hierarchy, 5–65
 - sourcefilter channel option, 33–109
 - systemfilter MTA option, 39–223
- systemfilter MTA option, 39–223
- Tests
 - address, 5–21
 - address, :aindex, 5–21
 - address, :anychild, 5–38
 - address, :comment, 5–21
 - address, :detail, 5–21
 - address, :display, 5–21
 - address, :index, 5–23
 - address, :last, 5–23
 - address, :mime, 5–38
 - address, :raw, 5–18, 5–21
 - address, :text, 5–18, 5–21
 - address, :user, 5–21
 - body, 5–21
 - body, :regex match type not supported, 5–61
 - currentdate, 5–23
 - Custom tests defined via mapping tables, 5–18
 - date, 5–23
 - duplicate, 5–18, 5–24
 - envelope, 5–26
 - envelope, conversiontag, 5–18, 5–27, 5–48
 - Environment, Custom items via \$+E
 - *_ACCESS flag, 5–28
 - exists, :anychild, 5–38
 - exists, :mime, 5–38
 - hasflag, 5–37
 - hasflag, :regex match type not supported, 5–61
 - header, :anychild, 5–38
 - header, :index, 5–23
 - header, :last, 5–23
 - header, :mime, 5–38
 - header, :raw, 5–18
 - header, :text, 5–18
 - header, defer_header_addition MTA option, 39–224
 - ihave, 5–37
 - importancetest, 5–52
 - Mapping tables, 5–63
 - memcache, 5–53
 - Message Store message expiration, 5–1
 - metermaid, 5–57
 - notify_method_capability, 5–40
 - regex, enable_sieve_regex MTA option, 39–229
 - Relational extension, 5–41
 - size, Operating on part vs. message, 5–38
 - size, Syntax of, 5–12
 - size, Units for values, 39–206
 - spamttest, 5–42
 - spamttest, :list, 5–42
 - spamttest, :percent, 5–42
 - string, 5–46
 - valid_ext_list, 5–29
 - valid_notify_method, 5–40
 - virustest, 5–42
 - virustest, :list, 5–42
- Timing of message discards and jettisons, 52–7
- translate action, 5–62
- URI encoding
 - :encodeurl, 5–40
- User-level
 - filter channel option, 33–109
 - ldap_domain_attr_filter MTA option, 39–148
 - ldap_filter MTA option, 39–129
 - ldap_hoh_filter MTA option, 39–97, 39–141
 - mailDomainSieveRuleSource LDAP attribute, 39–148
 - mailSieveRuleSource LDAP attribute, 39–129
 - mailSieveRuleSource LDAP attribute in Head-of-household entry, 39–97, 39–141
 - Sieve hierarchy, 5–65
- UTF-8 strings, 5–62
- vacation action
 - :addresses argument, ldap_autoreply_addresses MTA option, 39–128
 - :addresses argument, Vacation message not generated, 5–45
 - :days parameter, 5–45
 - :echo, 5–44
 - :echo argument, 47–8
 - :echo argument, mailAutoReplyMode attribute, 39–126
 - :headers, 5–44
 - :hours, 5–43

- :hours argument, ldap_autoreply_timeout MTA option, 39–129
- :noaddresses, 5–44
- :reply, 5–44
- :reply argument, 47–8
- :reply argument, mailAutoReplyMode attribute, 39–126
- :subject argument, mailAutoReplySubject attribute, 39–126
- Disabling, Initial configuration
- FROM_ACCESS mapping table, 5–44
- Disabling, Via `!` flag in FROM_ACCESS mapping, 5–44
- Disabling, Via address access mapping tables, 44–9
- Disabling, Via system-level novacation, 5–44
- Format of message, 47–8
- Frequency of vacation messages, autoreply_timeout_default MTA option, 39–65
- Logging of, 5–44
- max_vacations MTA option, 39–227
- Timing of response message generation, 47–4
- vacation_cleanup MTA option, 39–66
- vacation_maximum_timeout MTA option, 39–66, 39–102
- vacation_minimum_timeout MTA option, 39–66, 39–102
- vacation_template MTA option, 39–67
- vacation extension, 5–43
- vacation-seconds extension, 5–43
- valid_ext_list test, 5–29
- valid_notify_method test, 5–40
- variables extension, 5–46
 - :regex, 5–61
 - :regex match type tests, 5–47
 - Example, 5–49
 - FILTER_testname test results returned in, 5–63
 - Integer and list variable values, 5–48
 - Local to subroutine, 5–64
 - max_variables MTA option, 39–227
 - Namespaces not supported, 5–47
 - Periods not allowed in variable name, 5–47
 - Properties of external lists, 5–30, 5–47
 - set action, :encodeurl modifier, 5–40
 - Substitutions not allowed in body arguments, 5–46
 - Substitutions not supported in body test arguments, 5–21
- virusset action, 5–42
- virustest test, 5–42
 - :list, 5–42
- warn action, 5–63
 - log_filter MTA option, 39–232, 39–257
 - Syntax, 5–9
- sieve_received MTA option, 39–225
- sieve_redirect_add_resent MTA option
 - Default for redirect action, 5–41
- silentetrn channel option, 33–116
- single channel option, 33–57
 - Channel to a gateway system, 49–47
 - Effect via deliveryflags channel option, 33–108, 33–123
 - Pipe channels, 52–13, 52–16
- singlesignoff MSHTTP option, 29–8
- single_sys channel option, 33–57
 - Channel to a gateway system, 49–47
- slave channel option, 33–102
- slave_command Job Controller option, 42–14
- slave_debug channel option, 33–85
 - `$U` flag in PORT_ACCESS mapping table, 44–4
 - Force effect from address access mapping tables, 44–9
- os_debug MTA option, 39–75
- RESETDEBUG Archive option, 45–9
- SMPP relay
 - See SMS smpp_relay, 53–2
- SMPP server
 - See SMS smpp_server, 53–2
- SMS
 - charset
 - smsc_default_charset SMS gateway_profile option, 53–8
 - One-way
 - make_source_addresses_unique smpp_relay option, 53–10
 - Priority
 - sms_use_priority SMS gateway_profile option, 53–8
- SMS channels
 - Addresses
 - Example, 36–13, 36–14
 - ignore*encoding channel options, 33–47
- SMS gateway
 - Options, 53–2
 - debug, 7–21, 53–2
 - enable, 53–2
 - foreground, 53–3
 - history_file_directory, 53–3
 - history_file_flush_period, 53–3
 - history_file_flush_threshold, 53–3
 - history_file_mode, 53–3
 - history_file_rollover_period, 53–3
 - max_conns, 53–4
 - record_lifetime, 53–4

- thread_count_initial, 53-4
- thread_count_maximum, 53-4
- thread_stack_size, 53-4
- use_sms_priority, 53-8
- use_sms_privacy, 53-9
- Startup, 53-2
- SMS gateway_profile
 - Options, 53-4
 - email_body_charset, 53-5
 - email_body_charset, SMS text message body charset, 53-7
 - email_header_charset, 53-5
 - email_header_charset, Subject: header line, 53-7
 - from_domain, 53-5
 - in_re, 53-5
 - mta_channel, 53-5
 - parse_re_*, 53-6
 - route_to, 53-8
 - select_re, 53-8
 - smsc_default_charset, 53-8
 - text_to_subject, 53-4
 - text_to_subject, email_body_charset gets ignored, 53-5
 - text_to_subject, Interaction with parse_re_0 gateway_profile option, 53-7
- SMS options, 53-2
- SMS smpp_relay
 - Options, 53-9
 - backlog, 53-9
 - listen_addresses, 53-9
 - listen_receive_timeout, 53-9, 53-13
 - listen_transmit_timeout, 53-9, 53-13
 - make_source_addresses_unique, 53-9
 - max_conns, 53-10
 - server_host, 53-10
 - server_port, 53-10
 - server_receive_timeout, 53-10, 53-10
 - tcp_ports, 41-9, 41-9, 53-10
- SMS smpp_server
 - Options, 53-10
 - backlog, 53-10
 - esme_address_npi, 53-11
 - esme_address_range, 53-11
 - esme_address_ton, 53-11
 - esme_password, 53-12
 - esme_system_id, 53-12
 - esme_system_type, 53-12
 - listen_addresses, 53-12
 - listen_receive_timeout, 53-9, 53-13
 - listen_transmit_timeout, 53-9, 53-13
 - max_conns, 53-13
 - server_host, 53-13
 - server_port, 53-13
 - system_id, 53-13
 - tcp_ports, 41-9, 41-9, 53-13
- smsc_default_charset SMS gateway_profile option, 53-8
- SMTP
 - AUTH
 - alias_username_moderator_list alias option, 35-17
 - mail LDAP attribute, FROM_ACCESS mapping probe, 44-14
 - MSHTTP support for, 33-150
 - Requiring for mailing list postings, 39-132
 - SASL library code used, 35-5, 39-103
 - smtpauthpassword MSHTTP option, 29-6
 - smtpauthuser MSHTTP option, 29-6
 - Banner
 - BANNER_ADDITION TCP/IP-channel-specific option, 49-21
 - BANNER_PURGE_DELAY TCP/IP-channel-specific option, 49-21
 - CUSTOM_VERSION_STRING TCP/IP-channel-specific option, 49-23
 - fire away, 33-118
 - Host name, 33-80
 - Host name, BANNER_HOST TCP/IP-channel-specific option, 49-21
 - Host name, BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49-21
 - Logging and LOG_BANNER TCP/IP-channel-specific option, 49-27
 - MurkWorks, 33-118
 - STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49-36
 - Commands
 - ATRN, Safer than TURN, 33-132
 - AUTH, \$A input flag in AUTH_REWRITE mapping table, 33-33, 33-64, 33-148
 - AUTH, Adding authenticated sender to headers, 44-14
 - AUTH, authpassword channel option, 33-146
 - AUTH, authusername channel option, 33-146
 - AUTH, AUTH_ACCESS mapping \$Q flag, 49-38
 - AUTH, AUTH_PASSWORD TCP/IP-channel-specific option, 49-20
 - AUTH, AUTH_REWRITE mapping table, 33-32, 33-64, 33-148
 - AUTH, AUTH_USERNAME TCP/IP-channel-specific option, 49-20
 - AUTH, Channel switch with sasls witchchannel, 33-82, 33-154

AUTH, externalidentity channel option, 33–146
 AUTH, EXTERNAL_IDENTITY TCP/IP-channel-specific option, 49–20
 AUTH, Flag test in address access mapping tables, 44–9
 AUTH, implicitsaslexternal channel option, 33–151
 AUTH, logging of and log_auth MTA option, 39–252
 AUTH, mailAllowedServiceAccess effect, 49–52
 AUTH, mailUserStatus effect, 49–52
 AUTH, MSHTTP support for, 33–150
 AUTH, SASL error recorded in log_message_id field, 39–264
 AUTH, SASL library code used, 35–5, 39–103
 AUTH, SMTP/LMTP client use, 33–146, 49–20
 BDAT, 33–119
 BDAT, BURL interlaces with, 49–11
 Beginning of SMTP session, INITIAL_COMMAND TCP/IP-channel-specific option, 49–27
 BURL, imap_password MTA option, 39–68
 BURL, imap_username MTA option, 39–68
 Channel options, 33–116
 COMMAND_RECEIVE_TIME TCP/IP-channel-specific option, 49–22
 DATA, BURL replaces, 49–11
 DATA, DOT_TRANSMIT_TIME TCP/IP-channel-specific option, 49–26
 DATA, STATUS_DATA_RECEIVE_TIME TCP/IP-channel-specific option, 49–35
 DATA, STATUS_DATA_RECV_PER_ADDR_PER_BLK_TIME TCP/IP-channel-specific option, 49–35
 DATA, STATUS_DATA_RECV_PER_ADDR_TIME TCP/IP-channel-specific option, 49–35
 DATA, STATUS_DATA_RECV_PER_BLOCK_TIME TCP/IP-channel-specific option, 49–35
 DATA, Timeout with DATA_RECEIVE_TIME TCP/IP-channel-specific option, 49–23
 DATA, Timeout with DATA_TRANSMIT_TIME TCP/IP-channel-specific option, 49–23
 Disabling probe comments, 57–1
 EHLO, 33–118
 EHLO, \$E input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148
 EHLO, ehlokeywords submitproxy option, 27–14
 EHLO, ehlokeywords vdomain option, 27–14
 EHLO, Streaming of, 33–134
 EHLO/HELO name, Sieve filter test of, 5–28
 EHLO/HELO, BANNER_HOST TCP/IP-channel-specific option, 49–21
 EHLO/HELO, BANNER_REVERSE_HOST TCP/IP-channel-specific option, 49–21
 EHLO/HELO, HELLO_RECEIVE_TIME TCP/IP-channel-specific option, 49–26
 EHLO/HELO, Host name, 33–80
 EHLO/HELO, MAX_HELO_DOMAIN_LENGTH TCP/IP-channel-specific option, 49–30
 EHLO/HELO, SPF error and error_text_spf_ehlo_fail_4 MTA option, 39–165
 EHLO/HELO, SPF error and error_text_spf_ehlo_fail_5 MTA option, 39–165
 EHLO/HELO, SPF error and error_text_spf_ehlo_permerror_4 MTA option, 39–164
 EHLO/HELO, SPF error and error_text_spf_ehlo_permerror_5 MTA option, 39–164
 EHLO/HELO, SPF error and error_text_spf_ehlo_softfail_4 MTA option, 39–165
 EHLO/HELO, SPF error and error_text_spf_ehlo_softfail_5 MTA option, 39–165
 EHLO/HELO, SPF error and error_text_spf_ehlo_temperror_4 MTA option, 39–164
 EHLO/HELO, SPF error and error_text_spf_ehlo_temperror_5 MTA option, 39–164
 EHLO/HELO, STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
 ETRN, *etrn channel options, 33–116, 49–51
 ETRN, *sendetrn channel options, 33–131
 ETRN, allowetrn channel option, 49–51
 ETRN, ALLOW_ETRNS_PER_SESSION TCP/IP-channel-specific option, 49–18
 ETRN, Client's host name recorded in log_message_id field, 39–264
 ETRN, Disabled for SMTP SUBMIT, 33–119
 ETRN, ETRN_ACCESS mapping table, 33–117, 49–51, 49–51
 ETRN, Logging and the log_connection MTA option, 39–254

ETRN, Logging and the LOG_CONNECTION TCP/IP-channel-specific option, 49–27
 ETRN, Sending to trigger message transfer from remote systems, 49–51
 EXPN, alias_nonexpandable alias option, 35–15
 EXPN, Default handling for lists, 39–184
 EXPN, DISABLE_EXPAND TCP/IP-channel-specific option, 49–24
 EXPN, expandable LDAP attribute, 39–140
 EXPN, expandable_default channel option, 39–14
 EXPN, expandable_default MTA option, 39–184
 EXPN, expnallow channel option, 33–126
 EXPN, expndefault channel option, 33–126
 EXPN, expndisable channel option, 33–126
 EXPN, ldap_expandable MTA option, 39–140
 EXPN, Limited by posting access controls, 36–19
 EXPN, mgmanMemberVisibility LDAP attribute, 39–140
 EXPN, NONEXPANDABLE alias file named parameter, 35–33
 HELO, 33–118
 HELO, Streaming of, 33–134
 Logging bad commands, MAX_B_ENTRIES TCP/IP-channel-specific option, 49–29
 MAIL FROM,
 552_PERMANENT_ERROR_STRING TCP/IP-channel-specific option, 49–17
 MAIL FROM, Access control, 44–15
 MAIL FROM,
 ALLOW_TRANSACTIONS_PER_SESSION TCP/IP-channel-specific option, 49–20
 MAIL FROM, AUTH parameter, 33–154
 MAIL FROM, AUTH parameter and AUTH_ACCESS mapping table, 49–37
 MAIL FROM, AUTH parameter and AUTH_REWRITE mapping table, 33–33, 33–65, 33–149
 MAIL FROM, AUTH parameter and sasl*auth channel options, 33–154
 MAIL FROM, AUTH_REWRITE mapping table probe, 33–33, 33–64, 33–148
 MAIL FROM, DNS lookups, 33–129, 33–140
 MAIL FROM, ENVID parameter, 33–96, 33–131, 39–256
 MAIL FROM, from_access mapping table, 44–2, 44–13, 44–15
 MAIL FROM, implicitsaslexternal channel option, 33–151
 MAIL FROM, Limiting transactions accepted, 33–124
 MAIL FROM, mailDomainMsgMaxBlocks effect, 39–146
 MAIL FROM, MAIL_TRANSMIT_TIME TCP/IP-channel-specific option, 49–28
 MAIL FROM, MT-PRIORITY effect on notification generation, 33–97
 MAIL FROM, MTRK parameter, 33–91
 MAIL FROM, Null MX check with returnenvelope channel option, 33–99
 MAIL FROM, Null MX check with return_envelope MTA option, 39–156, 39–216
 MAIL FROM, ORCPT parameter recorded Original-recipient: header field, 33–63
 MAIL FROM, redirect Sieve action, 5–41
 MAIL FROM, RET parameter, 33–96, 33–131, 39–207, 39–213
 MAIL FROM, RET=HDRS, 39–207, 39–213
 MAIL FROM, Return-path: header field, 33–63
 MAIL FROM, Sieve filter evaluation, 5–67
 MAIL FROM, SOLICIT parameter, 33–124
 MAIL FROM, Spamfilter early verdict, 45–10
 MAIL FROM, SPF error and error_text_spf_fail_4 MTA option, 39–163
 MAIL FROM, SPF error and error_text_spf_fail_5 MTA option, 39–163
 MAIL FROM, SPF error and error_text_spf_permerror_4 MTA option, 39–163
 MAIL FROM, SPF error and error_text_spf_permerror_5 MTA option, 39–163
 MAIL FROM, SPF error and error_text_spf_softfail_4 MTA option, 39–164
 MAIL FROM, SPF error and error_text_spf_softfail_5 MTA option, 39–164
 MAIL FROM, SPF error and error_text_spf_temperror_4 MTA option, 39–163
 MAIL FROM, SPF error and error_text_spf_temperror_5 MTA option, 39–163
 MAIL FROM,
 STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
 MAIL FROM, Streaming of, 33–134
 MAIL FROM, Submission information string in certain *_access mapping table probes, 44–8, 44–8

MAIL FROM,
 TRANSACTION_LIMIT_RCPT_TO TCP/IP-channel-specific option, 49–36
 RCPT TO and VRFY,
 ALLOW_REJECTIONS_BEFORE_DEFERRAL TCP/IP-channel-specific option, 49–19
 RCPT TO,
 552_PERMANENT_ERROR_STRING TCP/IP-channel-specific option, 49–17
 RCPT TO, Access control, 44–2, 44–15
 RCPT TO,
 ALLOW_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option, 49–18
 RCPT TO, Limiting recipients accepted, 33–87, 33–120
 RCPT TO, MAIL_ACCESS mapping table, 44–2, 44–7
 RCPT TO, mail_access mapping table, 44–15
 RCPT TO, NOTIFY parameter, 33–96, 33–131
 RCPT TO, ORCPT parameter, 33–96, 33–131
 RCPT TO, ORIG_MAIL_ACCESS mapping table, 44–2, 44–7
 RCPT TO, orig_mail_access mapping table, 44–15
 RCPT TO, ORIG_SEND_ACCESS mapping table, 44–2, 44–7
 RCPT TO, orig_send_access mapping table, 44–15
 RCPT TO, RCPT_TRANSMIT_TIME TCP/IP-channel-specific option, 49–32
 RCPT TO,
 REJECT_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option, 49–32
 RCPT TO, RRVS parameter and alias_creation_date alias option, 35–12
 RCPT TO, RRVS parameter and checkrrvs channel option, 33–35, 33–118
 RCPT TO, RRVS parameter and ldap_creation_date MTA option, 39–151
 RCPT TO, RRVS parameter and ldap_domain_attr_creation_date MTA option, 39–151
 RCPT TO, SEND_ACCESS mapping table, 44–2, 44–7
 RCPT TO, send_access mapping table, 44–15
 RCPT TO, Sieve filter evaluation, 5–67
 RCPT TO, Sieve filter redirect-dsn extension, 5–41
 RCPT TO, Spamfilter early verdict, 45–10
 RCPT TO, SPF error and error_text_spf_fail_4 MTA option, 39–163
 RCPT TO, SPF error and error_text_spf_fail_5 MTA option, 39–163
 RCPT TO, SPF error and error_text_spf_permerror_4 MTA option, 39–163
 RCPT TO, SPF error and error_text_spf_permerror_5 MTA option, 39–163
 RCPT TO, SPF error and error_text_spf_softfail_4 MTA option, 39–164
 RCPT TO, SPF error and error_text_spf_softfail_5 MTA option, 39–164
 RCPT TO, SPF error and error_text_spf_temperror_4 MTA option, 39–163
 RCPT TO, SPF error and error_text_spf_temperror_5 MTA option, 39–163
 RCPT TO, STATUS_RCPT_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
 RCPT TO, Streaming of, 33–134
 RCPT TO,
 TRANSACTION_LIMIT_RCPT_TO TCP/IP-channel-specific option, 49–36
 RCPT TO, XAFLG parameter, 33–109, 33–124
 RCPT TO, XDFLG parameter, 33–109, 33–124
 RSET, Limiting transactions accepted, 33–125
 RSET, STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
 RSET, Streaming of, 33–134
 SAML FROM, DISABLE_SEND TCP/IP-channel-specific option, 49–25
 SAML FROM, Submission information string in certain *_access mapping table probes, 44–8
 SAML, error_text_send_remote_error MTA option, 39–158
 SAML, error_text_send_unknown_error MTA option, 39–159
 SEND FROM, DISABLE_SEND TCP/IP-channel-specific option, 49–25
 SEND FROM, Submission information string in certain *_access mapping table probes, 44–8
 SEND, error_text_send_remote_error MTA option, 39–158
 SEND, error_text_send_unknown_error MTA option, 39–159
 SOML FROM, DISABLE_SEND TCP/IP-channel-specific option, 49–25
 SOML FROM, Submission information string in certain *_access mapping table probes, 44–8
 STARTTLS, *tls* channel options, 33–83, 33–151
 STARTTLS, CLIENT_CERT_NICKNAME TCP/IP-channel-specific option, 49–22

STARTTLS, Flag test in address access mapping tables, 44–9

STARTTLS, Reattempting connection without TLS after failure, 49–35

STARTTLS, smtps MSHTTP option, 29–11

STARTTLS, TLS_ACCESS mapping table, 49–45

STATUS_RECEIVE_TIME TCP/IP-channel-specific option, 49–36

STATUS_TRANSMIT_TIME TCP/IP-channel-specific option, 49–36

TURN, 33–132

VERFY, *vrfy channel options, 33–125

VERFY, 552_PERMANENT_ERROR_STRING TCP/IP-channel-specific option, 49–17

VERFY, ALLOW_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option, 49–18

VERFY, HIDE_VERIFY TCP/IP-channel-specific option, 49–26

VERFY, REJECT_RECIPIENTS_PER_TRANSACTION TCP/IP-channel-specific option, 49–32

VERFY, vrfy* channel options, 33–134

XADR, \$V flag in PORT_ACCESS mapping table, 44–4

XADR, DISABLE_ADDRESS TCP/IP-channel-specific option, 49–23

XCIR, \$V flag in PORT_ACCESS mapping table, 44–4

XCIR, DISABLE_CIRCUIT TCP/IP-channel-specific option, 49–24

XGEN, \$V flag in PORT_ACCESS mapping table, 44–4

XGEN, DISABLE_GENERAL TCP/IP-channel-specific option, 49–25

XPEHLO, PORT_ACCESS mapping probe, 44–16

XPEHLO, PROXY_PASSWORD legacy configuration TCP/IP-channel-specific option, 49–31

XPEHLO, PROXY_PASSWORD TCP/IP-channel-specific option, 49–31

XPEHLO, smtpproxypassword option, 27–22, 39–218

XPEHLO, smtprelays SUBMIT Proxy option, 27–23

XSTA, \$V flag in PORT_ACCESS mapping table, 44–4

XSTA, DISABLE_STATUS TCP/IP-channel-specific option, 49–25

XUNAUTHENTICATE, Undoes sasls witchchannel effect, 33–83, 33–155

Connection

- Access control, 44–15
- PORT_ACCESS mapping table, 44–3

Debugging

- \$U flag in address *_ACCESS mapping table, 44–9
- \$U flag in PORT_ACCESS mapping table, 44–4
- debug_flush MTA option, 39–73, 39–172
- mm_debug MTA option, 39–74
- slave_debug channel option, 33–85

Delays in responses

- Force via address access mapping tables, 44–9

Delays on incoming SMTP sessions

- *_DELAY_* TCP/IP-channel-specific options, 49–34

DELIVERBY

- deliverbychannel channel option, 33–124
- deliverbymin channel option, 33–124
- log_deliver_by MTA option, 39–255

Disconnect

- ALLOW_SESSION_BLOCKS TCP/IP channel option, 49–19
- ALLOW_TRANSACTION_BLOCKS TCP/IP channel option, 49–20
- BURL_ACCESS forced disconnect, 49–9
- Forcing via disconnect* channel options, 33–87, 33–120
- Forcing via disconnecttransactionlimit channel option, 33–125

DSN

- Flag test in address access mapping tables, 44–9

EHLO announcement

- NO-SOLICITING, 33–124

EHLO/HELO name claimed by sending system

- Information included in certain *_access mapping table probes, 44–8

Error text

- Recipient access mapping table rejections, 44–8
- Set via address access mapping tables, 44–9

Error type

- Set via address access mapping tables, 44–9

Errors

- (bad authentication limit reached; disconnecting), 33–150
- 250 2.5.0 return address invalid/unroutable but accepted anyway, error_text_accepted_return_address MTA option, 39–166
- 252 2.5.0 Possible local address, 49–26

252 2.5.0 Possible remote address not checked., 49–27

421 4.7.0 Session recipient limit reached; disconnecting, 33–88, 33–122

421 4.7.0 Session rejection limit reached; disconnecting, 33–88, 33–122

450 4.0.0 Cannot find DIRECTORY value in options, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–9

450 4.0.0 Cannot find style value in options, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–9

450 4.0.0 Error opening archive options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–9

450 4.0.0 Error reading archive options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–9

450 4.0.0 Error reading ClamAV options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–4

450 4.0.0 Error reading ICAP options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–5

450 4.0.0 Error reading Milster options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–5

450 4.0.0 Error reading SpamAssassin options file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–7

450 4.0.0 No host specified in ClamAV option file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–4

450 4.0.0 No host specified in SpamAssassin option file, At MAIL FROM in JES MS 6.0 or JES MS 6.1, 45–7

450 4.1.8 invalid/host-not-in-DNS return address not allowed, 33–129, 33–141

450 4.1.8 invalid/host-not-in-DNS return address not allowed, error_text_mailfromdnsverify MTA option, 39–165

450 4.3.0 Cannot find DIRECTORY value in options, At MAIL FROM in JES MS 6.2 or later, 45–9

450 4.3.0 Cannot find style value in options, At MAIL FROM in JES MS 6.2 or later, 45–9

450 4.3.0 Error opening archive options file, At MAIL FROM in JES MS 6.2 or later, 45–9

450 4.3.0 Error reading archive options file, At MAIL FROM in JES MS 6.2 or later, 45–9

450 4.3.0 Error reading ClamAV options file, At MAIL FROM in JES MS 6.2 or later, 45–4

450 4.3.0 Error reading ICAP options file, At MAIL FROM in JES MS 6.2 or later, 45–5

450 4.3.0 Error reading Milster options file, At MAIL FROM in JES MS 6.2 or later, 45–5

450 4.3.0 Error reading SpamAssassin options file, At MAIL FROM in JES MS 6.2 or later, 45–7

450 4.3.0 No host specified in ClamAV option file, At MAIL FROM in JES MS 6.2 or later, 45–4

450 4.3.0 No host specified in SpamAssassin option file, At MAIL FROM in JES MS 6.2 or later, 45–7

450 4.3.0 source channel sieve filter access error, error_text_source_sieve_access MTA option, 39–166

450 4.3.0 source channel sieve filter authorization error, error_text_source_sieve_authorization MTA option, 39–166

450 4.3.0 source channel sieve filter syntax error:, error_text_source_sieve_syntax MTA option, 39–166

450 4.3.1 insufficient free queue space available, error_text_insufficient_queue_space MTA option, 39–166

450 4.5.1 permanent error in SPF verification of MAIL FROM domain (domain-name), 33–145, 39–242

450 4.5.1 SPF verification of MAIL FROM domain (domain-name) failed, 33–145

450 4.5.1 SPF verification of MAIL FROM domain (domain-name) failed (soft), 33–145

450 4.5.1 SPF verification of MAIL FROM domain (domain-name) failed: spf-explanation, 33–145

450 4.5.1 temporary error in SPF verification of MAIL FROM domain (domain-name), 33–145

450 4.5.3 number of transactions exceeds allowed maximum, 33–124

450 4.5.3 number of transactions exceeds allowed maximum, error_text_transaction_limit_exceeded, 39–166

450 4.7.0 Maximum number of commands exceeded, 33–88, 33–122

450 4.7.0 Maximum session time of <n> minutes has been exceeded, 49–33

450 4.7.0 Maximum transaction time of <n> minutes has been exceeded,

TRANSACTION_TIME TCP/IP-channel-specific option, 49–37

450 4.7.0 Session bad recipient limit reached; disconnecting, 33–88, 33–122

450 4.7.0 Session recipient limit reached; disconnecting, Issued prior to JES MS 6.3, 33–88, 33–122

450 4.7.0 Session rejection limit reached; disconnecting, Issued prior to JES MS 6.3, 33–88, 33–122

450 4.7.0 Session transaction limit reached; disconnecting, 33–125

450 4.7.1 filtering/scanning error, 39–250

451 4.3.0 SPF verification failed, 33–143

451 4.3.0 SPF verification failed: explanation-text, 33–143

451 4.4.3 Permanent error in SPF verification of HELO domain, 33–143, 39–241

451 4.4.3 Permanent error in SPF verification of MAIL FROM domain, 33–144

451 4.4.3 SPF verification failed, 33–144

451 4.4.3 SPF verification failed (soft), 33–145

451 4.4.3 SPF verification failed: explanation-text, 33–144

451 4.4.3 Temporary error in SPF verification of HELO domain, 33–143

451 4.4.3 Temporary error in SPF verification of MAIL FROM domain, 33–144

451 4.4.5 Error ending envelope - Too many recipients specified for this message, 33–87, 33–121

451 4.4.5 Error writing message temporaries, 39–166

451 4.4.5 error writing message temporary file, `error_text_temporary_write_error` MTA option, 39–166

451 4.5.2 Verification blocked; too many operations performed, 49–18

451 4.5.3 No more transactions allowed, 49–20, 49–37

451 4.5.3 too many recipients specified, 33–87, 33–121

451 4.5.3 Too many recipients specified, 49–18, 49–32

451 4.5.3 Too many rejections; try again later, 33–87, 33–121, 49–19

451 4.5.3 Transaction blocked; too many recipients specified, 49–19, 49–32

451 4.5.3 Verification blocked; too many operations performed, 49–32

451 4.5.3 Verification blocked; too many rejections, 49–19

451 4.7.1 filtering/scanning error, 39–250

451 4.7.1 filtering/scanning error, `error_text_spamfilter<N>_error` MTA options, 39–162

451 4.7.23 SPF verification of EHLO/HELO domain soft failed, `error_text_spf_ehlo_softfail_4` MTA option, 39–165

451 4.7.23 SPF verification of MAIL FROM domain soft failed (<domain>), `error_text_spf_softfail_4` MTA option, 39–164

451 4.7.24 temporary error in SPF verification of EHLO/HELO domain, `error_text_spf_ehlo_temperror_4` MTA option, 39–164

451 4.7.24 temporary error in SPF verification of MAIL FROM domain (<domain>), `error_text_spf_temperror_4` MTA option, 39–163

451 5.7.23 SPF verification of EHLO/HELO domain failed, `error_text_spf_ehlo_fail_4` MTA option, 39–165

451 5.7.23 SPF verification of MAIL FROM domain failed (<domain>), `error_text_spf_fail_4` MTA option, 39–163

451 5.7.24 permanent error in SPF verification of EHLO/HELO domain, `error_text_spf_ehlo_permerror_4` MTA option, 39–164

451 5.7.24 permanent error in SPF verification of MAIL FROM domain (<domain>), `error_text_spf_permerror_4` MTA option, 39–163

452 4.0.0 temporary error returned by alias expansion, `error_text_alias_temp` MTA option, 39–158

452 4.0.0 temporary error returned by alias expansion: address, 39–124

452 4.2.0 list is currently reserved and locked, `error_text_alias_locked` MTA option, 39–158

452 4.2.1 cannot reenqueue while still held, 52–11

452 4.2.1 cannot reenqueue while still held, `error_text_still_held` MTA option, 39–162

452 4.2.1 group temporarily disabled, `error_text_inactive_group` MTA option, 39–162

452 4.2.1 mailbox temporarily disabled, `error_text_inactive_user` MTA option, 39–162

452 4.2.2 user over quota; cannot receive new mail, `error_text_over_quota` MTA option, 39–161

452 4.2.3 too many recipients specified, `error_text_recipient_over` MTA option, 39–161

452 4.2.3 too many recipients specified, Issued prior to JES MS 6.3, 33–87, 33–121

452 4.3.0 filtering/scanning error, 39–250

452 4.3.4 message exceeds disk space available at this time, `error_text_insufficient_disk` MTA option, 39–161

452 4.5.0 error opening file/URL referenced by alias, `error_text_alias_fileerror` MTA option, 39–158

452 4.5.0 nonexistent file referenced by alias, `error_text_alias_fileexist` MTA option, 39–158

452 4.5.1 permanent error in SPF verification of MAIL FROM domain (domain-name), 39–242

452 4.5.1 Verification blocked; too many rejections, 33–89, 33–123

452 4.5.2 Verification blocked; too many bad addresses., Issued prior to JES MS 6.3, 33–89, 33–123

452 4.5.2 Verification blocked; too many operations performed, Issued prior to JES MS 6.3, 49–18

452 4.5.3 No more transactions allowed., Issued prior to JES MS 6.3, 49–20, 49–37

452 4.5.3 Too many recipients specified, Issued prior to JES MS 6.3, 49–18

452 4.5.3 Too many recipients specified., Issued prior to JES MS 6.3, 49–32

452 4.5.3 Too many rejections; try again later, Issued prior to JES MS 6.3, 33–87, 33–121

452 4.5.3 Too many rejections; try again later., Issued prior to JES MS 6.3, 49–19

452 4.5.3 Transaction blocked; too many recipients specified., Issued prior to JES MS 6.3, 49–19, 49–32

452 4.5.3 Verification blocked; too many bad addresses., Issued prior to JES MS 6.3, 49–19

452 4.5.3 Verification blocked; too many operations performed., Issued prior to JES MS 6.3, 49–32

452 4.5.3 Verification blocked; too many rejections., Issued prior to JES MS 6.3, 33–87, 33–121

452 4.7.1 Cannot find DIRECTORY value in options, At RCPT TO, 45–9

452 4.7.1 Cannot find style value in options, At RCPT TO, 45–9

452 4.7.1 Error opening archive options file, At RCPT TO, 45–9

452 4.7.1 Error reading archive options file, At RCPT TO, 45–9

452 4.7.1 Error reading ClamAV options file, At RCPT TO, 45–4

452 4.7.1 Error reading ICAP options file, At RCPT TO, 45–5

452 4.7.1 Error reading Milter options file, At RCPT TO, 45–5

452 4.7.1 Error reading SpamAssassin options file, At RCPT TO, 45–7

452 4.7.1 filtering/scanning error, 39–250

452 4.7.1 No host specified in ClamAV option file, At RCPT TO, 45–4

452 4.7.1 No host specified in SpamAssassin option file, At RCPT TO, 45–7

452 4.7.1 sieve filter access error, `error_text_sieve_access` MTA option, 39–161

452 4.7.1 sieve filter syntax error, `error_text_sieve_syntax` MTA option, 39–161

452 unknown host or domain, `error_text_temporary_failure` MTA option, 39–161

454 4.7.0 Authentication server unavailable, 49–53

458 4.7.1 ETRN session limit reached., `ALLOW_ETRNS_PER_SESSION` TCP/IP-channel-specific option, 49–18

459 4.5.0 Cannot start delivery on channel - access denied, 33–117, 49–51

459 4.5.0 site-supplied-error-text, 49–52

459 4.5.2 Cannot resolve name to SMTP channel, 33–116, 33–117

5.7.0 invalid originator address used, 33–34, 33–65, 33–149

5.7.1, 5–29

500 5.5.2 Permanent error in SPF verification of HELO domain, 33–143, 39–241

500 5.5.2 Temporary error in SPF verification of HELO domain, 33–143

501 5.5.0 Argument to EHLO is too long., `MAX_HELO_DOMAIN_LENGTH` TCP/IP-channel-specific option, 49–30

501 5.5.0 Argument to HELO is too long., `MAX_HELO_DOMAIN_LENGTH` TCP/IP-channel-specific option, 49–30

501 5.5.0 Argument to LHLO is too long., `MAX_HELO_DOMAIN_LENGTH` TCP/IP-channel-specific option, 49–30

501 5.5.0 Invalid input, 49–53

501 5.5.4 Future RRRVS value is not allowed, 33–35, 33–118

501 5.5.4 Mandatory MT-PRIORITY parameter is missing, 33–106, 33–131

501 5.5.4 MT-PRIORITY can only appear once, 33–106, 33–130

501 5.5.4 MT-PRIORITY value out of range, 33–106, 33–131

501 5.5.4 RRVS can only appear once, 33–35, 33–118

501 5.5.4 System name parameter is missing, 33–116

501 5.7.0 Cannot decode BASE64, 49–53

503 5.5.0 BURL illegal on LMTP port., 49–7

503 5.5.0 Proxy support is not enabled., 27–22, 39–218, 49–31

503 5.5.0 XUNAUTHENTICATE illegal on LMTP port, 33–83, 33–155

503 5.5.0 XUNAUTHENTICATE used while transaction is in progress, 33–83, 33–155

503 5.5.0 XUNAUTHENTICATE used while unauthenticated, 33–83, 33–155

503 5.5.1 BURL has not been enabled., 49–7

503 5.5.1 BURL only allowed on submission port., 49–8

503 5.5.1 ETRN not allowed on submission port, 33–116

503 5.7.1 FUTURERELEASE extension not available., 49–12, 49–13

503 5.7.1 RRVS extension not available, 33–35, 33–118

504 5.5.4 Unrecognized authentication type, 49–53

504 5.7.4 FUTURERELEASE extension not available., 49–12

504 5.7.4 FUTURERELEASE is a SUBMIT extension; it cannot be used in SMTP., 49–13

504 5.7.4 MT-PRIORITY extension not available, 33–106, 33–131

521 5.1.10 host/domain does not accept mail, 39–166

523 5.7.10 Encryption needed to use mechanism, 49–53

524 5.7.11 Password expired, has to be reset, 49–53

525 5.7.13 Account disabled, 49–52

530 5.7.0 No AUTH command has been given., 33–150

530 5.7.0 No STARTTLS command has been given., 33–84, 33–152

530 5.7.1 solicitations of this type are not allowed, error_text_nosolicit MTA option, 39–162

530 5.7.1 you are not allowed to use this list, error_text_alias_auth MTA option, 39–158

530 unknown host or domain, error_text_permanent_failure MTA option, 39–161

533 5.7.1 Access denied to specified URL., 49–9

535 5.7.8 Authorization failure, 49–52

535 5.7.8 Bad username or password, 49–52

535 5.7.8 SMTP proxy authentication check failed., 27–22, 39–218, 49–32

550 4.2.1 group temporarily disabled, error_text_inactive_group MTA option, 39–162

550 4.2.1 mailbox temporarily disabled, error_text_inactive_user MTA option, 39–162

550 4.2.2 user over quota; cannot receive new mail, error_text_over_quota MTA option, 39–161

550 4.2.3 too many recipients specified, Issued prior to JES MS 6.3, 33–87, 33–121

550 4.5.0 error opening file/URL referenced by alias, error_text_alias_fileerror MTA option, 39–158

550 4.5.0 nonexistent file referenced by alias, error_text_alias_fileexist MTA option, 39–158

550 4.5.3 too many recipients specified, 33–87, 33–121

550 4.7.23 SPF verification of EHLO/HELO domain soft failed, error_text_spf_ehlo_softfail_5 MTA option, 39–165

550 4.7.23 SPF verification of MAIL FROM domain soft failed (<domain>), error_text_spf_softfail_5 MTA option, 39–164

550 4.7.24 temporary error in SPF verification of EHLO/HELO domain, error_text_spf_ehlo_temperror_5 MTA option, 39–164

550 4.7.24 temporary error in SPF verification of MAIL FROM domain (<domain>), error_text_spf_temperror_5 MTA option, 39–163

550 5.1.1 unknown or illegal alias, error_text_unknown_alias MTA option, 39–158

550 5.1.1 unknown or illegal user, error_text_unknown_user MTA option, 39–158

550 5.1.2 unknown host or domain, error_text_unknown_host MTA option, 39–158

550 5.1.6 group no longer on server, error_text_deleted_group MTA option, 39–162

550 5.1.6 recipient no longer on server, error_text_deleted_user MTA option, 39–162

550 5.1.7 invalid/unroutable return address not allowed, `error_text_invalid_return_address` MTA option, 39–166

550 5.1.8 invalid/host-not-in-DNS return address not allowed, 33–129, 33–141

550 5.1.8 invalid/host-not-in-DNS return address not allowed, `error_text_mailfromdnsverify` MTA option, 39–165

550 5.1.8 invalid/no-such-user return address, `error_text_unknown_return_address` MTA option, 39–166

550 5.2.1 alias disabled; cannot receive new mail, `error_text_disabled_alias` MTA option, 39–161

550 5.2.1 group disabled; cannot receive new mail, `error_text_disabled_group` MTA option, 39–162

550 5.2.1 user disabled; cannot receive newmail, `error_text_disabled_user` MTA option, 39–161

550 5.2.3 channel limit of <n> kilobytes on message size exceeded, `error_text_block_over` MTA option, 39–159

550 5.2.3 channel limit of <n> lines on message length exceeded, `error_text_line_over` MTA option, 39–159

550 5.2.3 list limit of <n> kilobytes on message size exceeded, `error_text_list_block_over` MTA option, 39–159

550 5.2.3 list limit of <n> lines on message length exceeded, `error_text_list_line_over` MTA option, 39–160

550 5.2.3 user limit of <n> kilobytes on message size exceeded, `error_text_user_block_over` MTA option, 39–160

550 5.2.3 user limit of <n> lines on message length exceeded, `error_text_user_line_over` MTA option, 39–160

550 5.2.4 alias failed to expand to any valid addresses, `error_text_empty_alias` MTA option, 39–162

550 5.3.4 a message <n> lines long exceeds the line limit of <x> lines computed for this transaction, `error_text_message_too_long`, 39–160

550 5.3.4 a message size of <n> kilobytes exceeds the size limit of <x> kilobytes computed for this transaction, `error_text_message_too_large` MTA option, 39–160

550 5.5.0 permanent error in SPF verification of MAIL FROM domain (domain-name), 33–145, 39–241

550 5.5.0 SPF verification of MAIL FROM domain (domain-name) failed, 33–145

550 5.5.0 SPF verification of MAIL FROM domain (domain-name) failed (soft), 33–145

550 5.5.0 SPF verification of MAIL FROM domain (domain-name) failed: spf-explanation, 33–145

550 5.5.0 temporary error in SPF verification of MAIL FROM domain (domain-name), 33–145

550 5.5.1 ETRN has been disabled, 33–117

550 5.5.2 Permanent error in SPF verification of MAIL FROM domain, 33–144

550 5.5.2 Temporary error in SPF verification of MAIL FROM domain, 33–144

550 5.5.5 do not know how to SEND/SAML, `error_text_send_remote_error` MTA option, 39–159

550 5.6.0 lines longer than SMTP allows encountered; message rejected, 33–133

550 5.6.1 no protocol to SEND/SAML, `error_text_send_remote_error` MTA option, 39–158

550 5.7.0 Message priority outside currently allowed range, 33–106, 33–131

550 5.7.0 XADR command has been disabled., `DISABLE_ADDRESS` TCP/IP-channel-specific option, 49–24

550 5.7.0 XCIR command has been disabled., `DISABLE_CIRCUIT` TCP/IP-channel-specific option, 49–24

550 5.7.0 XGEN command has been disabled., `DISABLE_GENERAL` TCP/IP-channel-specific option, 49–25

550 5.7.0 XSTA command has been disabled., 49–25

550 5.7.1 ETRN session limit reached., `ALLOW_ETRNS_PER_SESSION` TCP/IP-channel-specific option, 49–18

550 5.7.1 Initial access check failure, 44–14

550 5.7.1 Solicitation check failure on `SOLICIT=`, 35–19

550 5.7.1 solicitations of this type are not allowed, `error_text_nosolicit` MTA option, 39–162

550 5.7.1 SPF verification failed, 33–143, 33–144

550 5.7.1 SPF verification failed (soft), 33–145

550 5.7.1 SPF verification failed: explanation-text, 33–143, 33–145

550 5.7.1 SRS/MUL address has a bad hash value, `error_text_srs_badhash` MTA option, 39–163

550 5.7.1 SRS/MUL address has timed out, `error_text_srs_timeout` MTA option, 39–163

550 5.7.1 unknown host or domain, Recipient *`_ACCESS` mapping tables, 44–8

550 5.7.1 you are not allowed to use this address, `error_text_access_failure` MTA option, 39–158

550 5.7.1 you are not allowed to use this address, Recipient *`_ACCESS` mapping tables, 44–8

550 5.7.1 you are not allowed to use this list, `error_text_alias_auth` MTA option, 39–158

550 5.7.15 account information on file is older than actual user account, 33–36, 33–119

550 5.7.17 account information on file is older than actual user account, `error_text_wrong_account`, 39–161

550 5.7.18 domain owner has changed, 33–36, 33–119

550 5.7.18 domain owner has changed, `error_text_wrong_domain` MTA option, 39–161

550 5.7.2 EXPN command has been disabled, 33–126

550 5.7.2 EXPN command has been disabled, `DISABLE_EXPAND` TCP/IP-channel-specific option, 49–24

550 5.7.2 SEND/SAML/SOML commands have been disabled., 49–25

550 5.7.23 SPF verification of EHLO/HELO domain failed, `error_text_spf_ehlo_fail_5` MTA option, 39–165

550 5.7.23 SPF verification of MAIL FROM domain failed (<domain>), `error_text_spf_fail_5` MTA option, 39–163

550 5.7.24 permanent error in SPF verification of EHLO/HELO domain, `error_text_spf_ehlo_permerror_5` MTA option, 39–164

550 5.7.24 permanent error in SPF verification of MAIL FROM domain (<domain>), `error_text_spf_permerror_5` MTA option, 39–163

550 5.7.26 host/domain does not accept mail, 39–166

550 unknown host or domain, `error_text_permanent_failure` MTA option, 39–161

552, `552_PERMANENT_ERROR_STRING` TCP/IP-channel-specific option, 49–17

553 5.1.0 8bit characters in address not allowed in this context, `error_text_disallowed_8bit` MTA option, 39–161

553 5.1.3 illegal 8bit characters in address, `error_text_illegal_8bit` MTA option, 39–161

553 5.1.3 Syntax error in SRS/MUL address, `error_text_srs_syntax` MTA option, 39–162

553 5.1.4 duplicate/ambiguous directory match, `error_text_duplicate_addrs` MTA option, 39–162

554 5.6.0 Error writing message - message is missing required recipient header fields, 33–39, 33–74, 39–59

554 5.6.0 lines longer than SMTP allowed encountered; message rejected, `error_text_smtp_lines_too_long` MTA option, 39–166

554 5.6.0 message contains unnegotiated 8bit, 33–53, 33–126

554 5.6.0 message contains unnegotiated 8bit, `error_text_unnegotiated_eightbit` MTA option, 39–166

554 5.6.0 message is missing required Date: header field, 33–120

555 5.5.2 Unrecognized MT-PRIORITY parameter value, 33–106, 33–131

555 5.5.4 Mandatory RRVS parameter is missing, 33–35, 33–118

555 5.5.4 Unrecognized RRVS parameter value , 33–35, 33–118

5yz error response at end of DATA due to Sieve refuse, 5–67

Maximum session data limit of <x>K octets has been exceeded, 49–19

Maximum transaction data limit of <x>K octets has been exceeded, 49–20

Notification messages generated by remote clients, 47–2

Syntax of, 39–157

Syntax of, `ldap_reject_text` MTA option, 39–132

Text of, `error_text_*` MTA options, 39–157

US-ASCII character set, 39–157

Extended error code

Set via address access mapping tables, 44–9

Extensions

AUTH, A modifier in MTA message transaction log entries, 55–4

AUTH, Sieve filter access to, 5–27

BDAT, Enabled by `binaryserver` even in absence of `chunkingserver`, 33–117

BINARYMIME, 33–117

BINARYMIME, B modifier in MTA message transaction log entries, 55–4
 BURL, submituser IMAP option, 21–4
 CHUNKING, 33–119
 CHUNKING, BURL, 49–11
 CHUNKING, C modifier in MTA message transaction log entries, 55–4
 DELIVERBY, deliverbychannel channel option, 33–124
 DELIVERBY, deliverbymin channel option, 33–124
 DELIVERBY, log_deliver_by MTA option, 39–255
 DSN, 33–96, 33–131
 DSN, Sieve filter redirect-dsn extension, 5–41
 DSN, Sieve filter setnotify and setreturn extensions, 5–61
 EHLO, 33–118
 EHLO, E modifier in MTA message transaction log entries, 55–4
 ETRN, Disabled for SMTP SUBMIT, 33–119
 ETRN, Sending to trigger message transfer from remote systems, 49–51
 FUTURERELEASE, 49–12
 FUTURERELEASE, Compared to Deferred-delivery: header line, 33–103
 FUTURERELEASE, log_futurerelease MTA option, 39–261
 Message Tracking, tracking* channel options, 33–91
 MT-PRIORITY, *backoff channel options, 33–101
 MT-PRIORITY, Effect on delivery retry frequency, 33–101
 MT-PRIORITY, Effect on MTA processing, 42–2
 MT-PRIORITY, Effect on timing of generation of notification messages, 33–97
 MT-PRIORITY, include_mtpriority MTA option, 39–191
 MT-PRIORITY, Job Controller delivery execution window, 42–16
 MT-PRIORITY, Mapping table probes, 39–191
 MT-PRIORITY, message_save_copy_flags MTA option, 39–198
 MT-PRIORITY, mtpriorities* channel options, 33–106, 33–130
 MT-PRIORITY, mtpriority_policy MTA option, 39–219
 MT-PRIORITY, Overrides size-based priority adjustment MTA options, 33–114, 39–209, 39–219
 MT-PRIORITY, Rewrite rule access to, 34–33
 MT-PRIORITY, setmtpriority Sieve action, 5–18
 MT-PRIORITY, Sieve filter access to, 5–27
 MT-PRIORITY, vnd.oracle.mt-priority Sieve filter environment item, 5–27
 NO-SOLICITING, alias_nosolicit alias option, 35–19
 NO-SOLICITING, destinationnosolicit channel option, 33–124
 NO-SOLICITING, error_text_nosolicit MTA option, 39–162, 39–162
 NO-SOLICITING, ldap_domain_attr_nosolicit MTA option, 39–147
 NO-SOLICITING, ldap_nosolicit MTA option, 39–119
 NO-SOLICITING, NOSOLICIT alias file named parameter, 35–36
 NO-SOLICITING, sourcenosolicit channel option, 33–124
 NOTARY, Groups vs. mailing lists, 36–18
 NOTARY, log_notary MTA option, 39–265
 NOTIFY, mailDomainMsgMaxBlocks effect, 39–146
 Operation Type, Sieve filter access to, 5–28
 Operation Type, vnd.oracle.operation-type Sieve filter environment item, 5–28
 PIPELINING, Q modifier in MTA message transaction log entries, 55–4
 PIPELINING, streaming channel option, 33–134
 RRV5, alias_creation_date alias option, 35–12
 RRV5, checkrrvs channel option, 33–35, 33–118
 RRV5, CREATION_DATE alias file named parameter, 35–31
 RRV5, ldap_creation_date MTA option, 39–151
 RRV5, ldap_domain_attr_creation_date MTA option, 39–151
 SIZE, 33–113
 SIZE, error_text_block_over MTA option, 39–159
 SIZE, include_mtpriority MTA option, 39–192
 SIZE, Interaction with MT-PRIORITY, 39–192
 SIZE, Mapping table probes, 39–192
 STARTTLS, CLIENT_CERT_NICKNAME TCP/IP-channel-specific option, 49–22
 STARTTLS, S modifier in MTA message transaction log entries, 55–4
 XADR, \$V flag in PORT_ACCESS mapping table, 44–4

- XCIR, \$V flag in PORT_ACCESS mapping table, 44-4
- XCLIENT, *xclient* channel options, 33-76, 33-132, 33-153
- XGEN, \$V flag in PORT_ACCESS mapping table, 44-4
- XLOOP, 33-128
- XPEHLO, PROXY_PASSWORD TCP/IP-channel-specific option, 49-31
- XSTA, \$V flag in PORT_ACCESS mapping table, 44-4
- XUNAUTHENTICATE, Undoes saslswitchchannel effect, 33-83, 33-155
- Illegally long lines, 33-133
- Line length, 33-47
- Line terminator, 33-128
- Line terminators
 - Channel options, 33-116
- msprobe probe of, 10-2
- Options
 - See also TCP/IP channels, Options, 49-16
- ORCPT
 - access_orcpt, 44-8
 - Included in certain *_access mapping table probes, 44-8
- Performance
 - REUSE_TIMED_OUT_TRANSFERS TCP/IP-channel-specific option, 49-33
- Pipelining and streaming
 - streaming channel option, 33-134
- Polling for messages, 49-51
- Server
 - See TCP/IP channels, 49-2
- Server log file name
 - logfilename Dispatcher service option, 41-5
- SSL without STARTTLS
 - SSL_CLIENT TCP/IP-channel-specific option, 49-34
- Success responses
 - 220 2.5.0 Session unauthenticated, 33-83, 33-155
 - 250 2.0.0 message accepted for list expansion processing, error_text_receipt_it MTA option, 39-162
 - 250 2.3.0 actual-mtpriority MT-PRIORITY value used; original value requested-mtpriority, 33-106, 33-130
 - 250 2.5.0 Prior aborted transfer used, 49-33
- smtp channel option, 33-127
- SMTP over TCP/IP channels
 - See TCP/IP channels, 49-2
- SMTP relay blocking, 49-48
- SRS, 49-50
- SMTP SUBMIT
 - BURL extension
 - MTA options, 39-68
 - BURL Extension
 - U modifier in MTA message transaction log entries, 55-4
 - Date: header, 33-119
 - ETRN disabled, 33-119
 - Extensions
 - BURL, 49-7
 - BURL, U modifier in MTA message transaction log entries, 55-4
 - FUTURERELEASE, futurerelease channel option, 33-104, 33-126
 - msprobe probe of, 10-2
 - Options
 - See also TCP/IP channels, Options, 49-16
 - See also SMTP, 49-6
 - Server, 49-6
 - Server log file name
 - logfilename Dispatcher service option, 41-5
 - Third party submission
 - AUTH_ACCESS mapping, 49-40, 49-41
- SMTP SUBMIT Proxy
 - Options
 - clientlookup, 27-9
 - smtpproxypassword, 27-22, 39-218
 - smtpauthpassword MSHTTP option, 29-6
 - smtpauthuser MSHTTP option, 29-6
 - smtpport MSHTTP option, 29-10
 - smtpport MSHTTP option, 29-11
 - smtpproxypassword MMP option, 27-22, 39-218
 - smtpproxypassword MTA option, 27-22, 39-218
 - smtpproxypassword SMTP SUBMIT Proxy option, 27-22, 39-218
 - smtprelays SUBMIT Proxy option, 27-23
 - smptpls MSHTTP option, 29-11
 - smtp_cr channel option, 33-127
 - smtp_crlf channel option, 33-127
 - smtp_crorlf channel option, 33-127
 - smtp_lf channel option, 33-127
 - snapshot
 - Enable scheduling of, 8-5
 - snapshot Message Store
 - crontab Scheduler task option, 8-4, 8-5
 - update
 - crontab Scheduler task option, 8-4, 8-5
 - snapshot options, 8-4
 - snapshotdirs Message Store deleted option, 16-26
 - snapshotdirs Message Store option, 16-8
 - snapshotpath Message Store option, 16-9
 - snapshotverify
 - Enable scheduling of, 8-5

- snapshotverify options, 8–5
- sndopr_priority MTA option, 39–249
 - Effect on log_sndopr MTA option, 39–249
 - held_sndopr MTA option, 39–221, 39–247
- SNMP, 1
- SNMP options, 60–1
 - cachetl, 60–1
 - contextname, 60–2
 - directoryscan, 60–2
 - enable, 60–1
 - enablecontextname, 60–2
 - listenaddr, 60–1
 - port, 60–1
 - registerindices, 60–3
 - servertimeout, 60–3
 - standalone, 60–3
- snmpd
 - standalone SNMP option, 60–3
- Socket Secure
 - See SOCKS, 33–141
- SOCKS
 - sockshost channel option, 33–142
 - sockspassword channel option, 33–142
 - socksport channel option, 33–142
 - socksusername channel option, 33–142
 - socksuserpassword channel option, 33–142
- Socks
 - Milter
 - Support removed in Messaging Server 8.0, 45–7, 45–13
- sockshost channel option, 33–142
- socksnoauth channel option, 33–142
- sockspassword channel option, 33–142
- socksport channel option, 33–142
- socksusername channel option, 33–142
- socksuserpassword channel option, 33–142
- softtokendir base option, 7–17
- Solaris system parameters, 56–3
 - datasize, 56–3
 - rlim_fd_max, 56–4
 - siffp_fd_max, 56–3
- sourceblocklimit channel option, 33–112
- sourcecommentinc channel option, 33–66
- sourcecommentmap channel option, 33–66
 - use_comment_strings MTA option, 39–198
- sourcecommentomit channel option, 33–66
- sourcecommentstrip channel option, 33–66
- sourcecommenttotal channel option, 33–66
- sourceconversiontag channel option, 33–55
- sourcefilter channel option, 33–109
 - error_text_source_sieve_access MTA option, 39–166
 - error_text_source_sieve_authorization MTA option, 39–166
 - error_text_source_sieve_syntax MTA option, 39–166
 - Message capture example, 5–35
 - Performance impact, 56–3
 - Sieve hierarchy, 5–65
- sourcenosolicit channel option, 33–124
- sourcepersonalinc channel option, 33–41, 33–76
- sourcepersonalmap channel option, 33–41, 33–76
 - use_personal_names MTA option, 39–201
- sourcepersonalomit channel option, 33–41, 33–76
- sourcepersonalstrip channel option, 33–41, 33–76
- sourceroute channel option, 33–35
- sourcesrs channel option, 33–29
- sourceurl MSHTTP option, 29–11
- source_channel Message Store archive option, 16–14
- spam MSHTTP feedback option, 29–14
- Spam/virus filter package integration, 45–1, 45–17
 - access_errors MTA option, 39–157
 - Address format
 - spamfilter*_final, 39–238
 - Address reversal, 35–50
 - aliasoptindetourhost channel option
 - aliasdetourhost_null_optin MTA option, 39–92
- Archiving
 - See Archive package integration, 45–9
- Brightmail
 - Default verdict, 39–237
 - Library image location, 39–234
 - spamfilterN_config_file options, 45–2
- Channel options, 33–115
- ClamAV
 - Configuration file location, 45–4
 - DEBUG ClamAV option, 45–4
 - FIELD ClamAV option, 45–4
 - HOST ClamAV option, 45–4
 - MESSAGE_BUFFER_SIZE ClamAV option, 45–4
 - MODE ClamAV option, 45–4
 - PORT ClamAV option, 45–4
 - SOCKS_HOST ClamAV option, 45–4
 - SOCKS_PASSWORD ClamAV option, 45–4
 - SOCKS_PORT ClamAV option, 45–4
 - SOCKS_USERNAME ClamAV option, 45–4
 - spamfilterN_config_file options, 45–4
 - TIMEOUT ClamAV option, 45–4
 - USE_INSTREAM ClamAV option, 45–4
 - VERDICT ClamAV option, 45–4
- Comparison of approaches, 45–1
- Configuration file location

- spamfilter*_config_file, 39–235
- Debugging
 - mm_debug MTA option, 39–74
- Early verdicts, 45–10
- Errors
 - access_errors MTA option, 39–157
- Headers passed to package
 - spamfilter*_includeheaders, 39–238
 - spamfilter*_received, 39–239
 - spamfilter*_returnpath, 39–239
- ICAP
 - Configuration file location, 45–5
 - Configuration options, 45–5
 - DEBUG ICAP option, 45–5
 - Debugging, 45–5
 - FIELD ICAP option, 45–5
 - HOST ICAP option, 45–5
 - Library image location, 39–234
 - MODE ICAP option, 45–5
 - PORT ICAP option, 45–5
 - SOCKS_HOST ICAP option, 45–5
 - SOCKS_PORT ICAP option, 45–5
 - SOCKS_USERNAME ICAP option, 45–5
 - spamfilterN_config_file options, 45–5
 - TIMEOUT ICAP option, 45–5
 - VERDICT ICAP option, 45–5
- IP reputation checks, 45–10
- Library location
 - spamfilter*_library, 39–234
- Milter, 45–11
 - .HELD files, 45–5
 - Actions supported, 45–12
 - Capabilities supported, 45–11
 - Configuration file location, 45–5
 - Configuration options, 45–5
 - CONNECT_TIMEOUT Milter option, 45–5
 - CONTEXT_EDITS, 45–5
 - DEBUG Milter option, 45–5
 - Debugging, 45–5
 - DEFER_MESSAGE_TRANSFER, 45–5
 - HOST Milter option, 45–5
 - IMMEDIATE_HEADER_MODIFICATIONS, 45–5
 - Library image location, 39–234
 - Macros supported, 45–12
 - Macros supported, SMFIF_SETSYMLIST milter action, 45–12
 - MAX_PREPEND_INDEX Milter option, 45–5
 - MILTER_MACROS mapping table, 45–14
 - OpenDKIM, MAX_PREPEND_INDEX milter spamfilter option, 45–13
 - OpenDKIM, Sieve counters, 5–51, 5–51
 - Per-recipient modification actions, 45–15
 - PER_RECIPIENT_ACTIONS, 45–5
 - PER_RECIPIENT_ACTIONS Milter option, 45–16
 - PORT Milter option, 45–5
 - PRESERVE_BREAKS Milter option, 45–5
 - QUARANTINE_ACTION Milter option, 45–5
 - RESETDEBUG Milter option, 45–5
 - Session reuse, 45–13
 - SESSION_INACTIVITY_TIMEOUT Milter option, 45–5, 45–13
 - SESSION_TIME Milter option, 45–13
 - SESSION_TIMEOUT Milter option, 45–5
 - Single recipient extension, 45–15
 - SMFIC_CONNECT, Early verdict, 45–10
 - SMFIF_SPECRCPT, 45–15
 - Socks connections, Options removed in Messaging Server 8.0, 45–13
 - SOCKS_HOST Milter option, Support removed in Messaging Server 8.0, 45–7, 45–13
 - SOCKS_PASSWORD Milter option, Support removed in Messaging Server 8.0, 45–7, 45–13
 - SOCKS_PORT Milter option, Support removed in Messaging Server 8.0, 45–7, 45–13
 - SOCKS_USERNAME Milter option, Support removed in Messaging Server 8.0, 45–7, 45–13
 - spamfilterN_config_file options, 45–5
 - spamfilterN_string_action MTA option, 39–240, 45–14
 - TIMEOUT Milter option, 45–5
 - TRANSACTIONS_PER_SESSION Milter option, 45–5, 45–13
 - USE_JETTISON Milter option, 45–5
 - USE_QUIT_NC Milter option, 45–5, 45–13
- MILTER_MACROS mapping table, 45–14
- MTA options, 39–233
 - access_errors, 39–156, 39–157
 - error_text_spamfilter*_error, 39–162
 - ldap_domain_attr_optin* MTA option, 39–146
 - ldap_optin*, 39–121
 - ldap_source_optin*, 39–118
 - optin_user_carryover, 39–99, 39–234
 - scan_channel, 39–170
 - scan_originator, 39–170
 - scan_recipient, 39–170
 - spamfilter*_action_*, 39–235
 - spamfilter*_config_file, 39–235
 - spamfilter*_final, 39–238
 - spamfilter*_includeheaders, 39–238
 - spamfilter*_library, 39–234
 - spamfilter*_null_action, 39–238
 - spamfilter*_null_optin, 39–235
 - spamfilter*_received, 39–239
 - spamfilter*_returnpath, 39–239

Special symbolic names, 3–1

SPF lookups

- Debugging
 - mm_debug MTA option, 39–74
- Fail
 - error_text_spf_ehlo_fail_4 MTA option, 39–165
 - error_text_spf_ehlo_fail_5 MTA option, 39–165
 - error_text_spf_fail_4 MTA option, 39–163
 - error_text_spf_fail_5 MTA option, 39–163
 - error_text_spf_softfail_5 MTA option, 39–164
- Fail (soft)
 - error_text_spf_ehlo_softfail_4 MTA option, 39–165
 - error_text_spf_ehlo_softfail_5 MTA option, 39–165
 - error_text_spf_softfail_4 MTA option, 39–164
- Forwarded messages, 39–244
- MTA options, 39–240
- Permanent DNS error
 - error_text_spf_ehlo_permerror_4 MTA option, 39–164
 - error_text_spf_ehlo_permerror_5 MTA option, 39–164
 - error_text_spf_permerror_4 MTA option, 39–163
 - error_text_spf_permerror_5 MTA option, 39–163
- spf* channel options, 33–143
- SPF_LOCAL mapping table avoids actual DNS lookups, 33–146
- Temporary DNS error
 - error_text_spf_ehlo_temperror_4 MTA option, 39–164
 - error_text_spf_ehlo_temperror_5 MTA option, 39–164
 - error_text_spf_temperror_4 MTA option, 39–163
 - error_text_spf_temperror_5 MTA option, 39–163
- spfhelo channel option, 33–143
- spfmailfrom channel option, 33–143
- spfnone channel option, 33–143
- spfrcptto channel option, 33–143
- spf_max_dns_queries MTA option, 39–244
- spf_max_recursion MTA option, 39–244
- spf_max_time MTA option, 39–244
- spf_smtp_status_fail MTA option, 39–240
 - spf* channel options, 33–144
- spf_smtp_status_fail_all MTA option, 39–241
 - spf* channel options, 33–144
- spf_smtp_status_permerror MTA option, 39–241
 - spf* channel options, 33–143
 - spfmailfrom channel option, 33–144
- spf_smtp_status_softfail MTA option, 39–242
 - spf* channel options, 33–145
- spf_smtp_status_softfail_all MTA option, 39–242
 - spf* channel options, 33–143
 - spfmailfrom channel option, 33–144
- spf_smtp_status_temperror MTA option, 39–243
 - spf* channel options, 33–143
 - spfmailfrom channel option, 33–144
- spoofemptymailbox POP Proxy option, 27–23
- spoofmessagefile POP Proxy option, 27–23
- spooftempfail POP Proxy option, 27–23
- spooldir MSHTTP option, 29–11
- SRS (Sender Rewriting Scheme)
 - Syntax errors
 - error_text_srs_syntax MTA option, 39–162
- SRS (Sender Rewriting Scheme)
 - *srs* channel options, 33–29
- Bad hash
 - error_text_srs_badhash MTA option, 39–163
- MTA options, 39–244, 39–245
 - token_char, 39–60, 39–246
- Relay blocking, 49–50
- Time out
 - error_text_srs_timeout MTA option, 39–163
- Used to work around SPF-caused problems in autoforwarding, 33–145
- srs_domain MTA option, 39–245
- srs_maxage MTA option, 39–245
 - error_text_srs_timeout MTA option, 39–163
- srs_secrets MTA option, 39–245
- SSL
 - checkoverssl S/MIME option, 30–5
- LDAP
 - ugldapport, 7–14
 - ugldapusessl, 7–15
- PAB
 - ldapusessl PAB option, 59–2
- proxyimapssl base option, 7–16
- sslcache size IMAP option, 21–12
- sslcache size MSHTTP option, 29–13
- sslcache size POP option, 22–4
- sslsourceurl MSHTTP option, 29–11
- uwcsslport MSHTTP option, 29–9

SSL/TLS

- \$T input flag in AUTH_REWRITE mapping table, 33–33, 33–64, 33–148
- capability_starttls Deploymap option, 14–2
- capability_starttls IMAP option, 21–8

Certificate

- sslnicknames base option, 7–17, 27–26
- sslnicknames option, 27–26

Channel options, 33–83, 33–151

Cipher suites

- ssladjustciphersuites base option, 7–11, 27–24
 - tlsv12enable base option, 7–18
- Client certificates
 - storeadmin mmp/imaproxy/popproxy/vdomain option, 27–27
 - storeadminpass mmp/imaproxy/popproxy/vdomain option, 27–27
- Compression option
 - sslcompress base option, 7–13
- Debugging
 - tls keyword in debugkeys option, 27–12
- Directory storing certificate and key files
 - ssldbpath option, 7–12
- enableslport ENS option, 61–1
- ENS
 - sslport ENS option, 61–1
- HTTP
 - plaintextmncipher MSHTTP option, 27–19, 29–12
- IMAP
 - enableslport IMAP option, 21–3
 - plaintextmncipher POP option, 21–12, 27–19
 - sslport IMAP option, 21–12
 - sslusessl IMAP option, 21–13
- LDAP
 - ldaprequiretls, 7–15
- ldapurl MMP option
 - ldaps: URL pointing to subtree of DIT, 27–17
- MSHTTP
 - enableslport MSHTTP option, 29–3
 - sslport MSHTTP option, 29–13
 - sslport MSHTTP sieve option, 29–23
 - sslusessl MSHTTP option, 29–13
- MTA options, 33–146, 39–217
 - plaintextmncipher, 27–19, 39–217
- PKIX verification of client certificates
 - sslpkix base option, 7–13
- plaintextmncipher IMAP option, 21–12, 27–19
- plaintextmncipher MSHTTP option, 27–19, 29–12
- plaintextmncipher MTA option, 27–19, 39–217
- plaintextmncipher option, 27–19
- plaintextmncipher POP option, 22–3, 27–19
- POP
 - enableslport POP option, 22–2
 - plaintextmncipher POP option, 22–3, 27–19
 - sslport POP option, 22–4
 - sslusessl POP option, 22–4
- Renegotiation
 - sslrenegotiate base option, 7–19
 - sslrequiresafenegotiate base option, 7–13
- restrictplainpasswords mmp/imaproxy/popproxy/vdomain option, 27–21
- Session cache directory
 - sslcachedir option, 7–3, 27–25
- SMTP
 - plaintextmncipher MTA option, 27–19, 39–217
- smtptls MSHTTP option, 29–11
- sslbacksideport IMAP Proxy and POP Proxy option, 27–25
- ssldblegacy base option, 7–12
- ssldbprefix base option, 7–13
- sslnicknames ENS option, 27–27, 61–2
- sslnicknames option, 27–26
- sslnicknames POP option, 22–4, 27–27
- sslport ENS option, 61–1
- sslport IMAP option, 21–12
- sslport MSHTTP option, 29–13
- sslport MSHTTP sieve option, 29–23
- sslport POP option, 22–4
- sslrenegotiate base option, 7–19
- sslrequiresafenegotiate base option, 7–13
- sslusessl option, 29–13
- sslv3enable base option, 7–13
- SSL_CLIENT TCP/IP-channel-specific option, 49–34
 - ssl_ports Dispatcher option, 41–9
 - ssl_ports tcp_listen option, 28–1
- STARTTLS_FAILURE_RECONNECT_DELAY TCP/IP-channel-specific option, 49–35
- tlsv12enable base option, 7–18
- TLS_ACCESS mapping table, 49–45
- tokenpass sectoken option, 13–1
- ssladjustciphersuites base option, 7–11, 27–24
- ssladjustciphersuites MMP/IMAP Proxy/POP Proxy/SUBMIT Proxy/vdomain option, 7–11, 27–24
- sslbacksideport IMAP Proxy and POP Proxy option, 27–25
- sslcachedir option, 7–3, 27–25
- sslcachesize IMAP option, 21–12
- sslcachesize MSHTTP option, 29–13
- sslcachesize POP option, 22–4
- sslcompress base option, 7–13
- ssldblegacy base option, 7–12
- ssldbpath base option, 7–12
- ssldbpath option
 - Precedence over sslcachedir option, 7–3, 27–25
- ssldbprefix base option, 7–13
- sslkeypasswdfile MMP/IMAP Proxy/POP Proxy/SUBMIT Proxy option
 - DELETED, 27–26
- sslnicknames base option, 7–17, 27–26
- sslnicknames ENS option, 27–27, 61–2
- sslnicknames IMAP option, 21–12, 27–26
- sslnicknames MSHTTP option, 27–26, 29–13

sslnicknames MTA option, 27–27, 39–218
 sslnicknames POP option, 22–4, 27–27
 sslpkix base option, 7–13
 sslport ENS option, 61–1
 sslport IMAP option, 21–12
 sslport MSHTTP option, 29–13
 sslport MSHTTP sieve option, 29–23
 sslport POP option, 22–4
 sslrenegotiate base option, 7–19
 sslrequiresafenegotiate base option, 7–13
 sslrootcacertsurl smime option, 30–2, 30–2
 sslsecmodfile MMP option
 DELETED, 27–27
 sslsourceurl MSHTTP option, 29–11
 sslusessl IMAP option, 21–13
 sslusessl MSHTTP option, 29–13
 sslusessl POP option, 22–4
 sslv3enable base option, 7–13
 ssl_ports Dispatcher option, 41–9, 41–9
 ssl_ports option
 Compared to maytls* channel options, 33–84,
 33–152
 ssl_ports tcp_listen option, 28–1
 SSO options, 31–1
 verifyurl, 31–1
 sso_enable MSHTTP option, 29–8
 sso_id MSHTTP option, 29–9
 sso_prefix MSHTTP option, 29–9
 SSR database
 ssr_database_url MTA option, 39–203
 ssr: URLs
 imta_ssr_database MTA Tailor option
 DELETED, 40–8
 MTA URL types, 1–4
 ssr_database_url MTA option, 39–203
 stacksize Dispatcher service option, 41–8
 standalone SNMP option, 60–3
 Standards
 Accept-language: header line
 See RFC 2068 (HTTP), 47–9
 draft-degener-sieve-editheader-00
 replaceheader Sieve action, 5–25
 draft-delany-nullmx-01.txt, 33–99, 39–156,
 39–216
 draft-ietf-appsawg-email-auth-codes-07, 33–144
 ISO 8601 duration format, 1–3
 ISO 8601 format, 1–3
 ISO 8601 P format, 1–3
 alias_deferred* alias options, 35–12
 Message tracking
 See RFCs 3885–3887, 48–1
 Null MX record entry
 draft-delany-nullmx-01.txt, 33–99, 39–156,
 39–216
 RFC 1123 (Requirements for Internet Hosts)
 Postmaster mailbox, 47–25
 RFC 1137 (Mapping Between Full RFC 822 and
 RFC 822 with Restricted Encoding), 33–40
 RFC 1154 (Encoding header field for internet
 messages)
 Encoding: header line, 33–46
 RFC 1413 (IDENT), 33–138
 RFC 1738 (Uniform Resource Locators (URL)),
 35–7
 Alias file LDAP URL alias encoding, 35–42
 URL character encoding issues, 37–14
 RFC 1766 (Tags for the Identification of
 Languages)
 See RFC 3066 (Tags for the Identification of
 Languages), 47–9
 RFC 1870 (ESMTP message size extension)
 Message size limits, 33–113
 RFC 1928 (SOCKS V5), 33–142
 RFC 1929 (Username/Password Authentication
 for SOCKS V5), 33–142
 RFC 2068 (Hypertext Transfer Protocol --
 HTTP/1.1), 47–9
 Accept-Language: definition,
 installedlanguages base option, 7–8
 RFC 2087 (IMAP4 QUOTA extension), 16–22
 RFC 2156 (MIXER: Mapping between X.400 and
 RFC 822/MIME)
 Deferred-delivery: header, 33–102
 Influence on MTA's Priority Assignment
 Policy, 39–219
 RFC 2197 (SMTP Service Extension for
 Command Pipelining)
 streaming channel option, 33–134
 RFC 2231 (MIME Parameter Value and Encoded
 Word Extensions: Character sets, Languages,
 and Continuations), 33–50, 33–54
 parameterformat* channel options, 33–50,
 33–54
 rfc2231compliant MSHTTP option, 29–5
 RFC 2254 (The String Representation of LDAP
 Search Filters)
 String search filter definition, 37–14
 RFC 2255 (LDAP URL Format), 35–7
 Alias file syntax, 35–41
 alias_urlN MTA option syntax, 35–6
 RFC 2369, 35–16, 35–34
 RFC 2369 (URLs for Mail List Commands
 through Message Headers), 39–139
 RFC 2595 (Using TLS with IMAP, POP3, and
 ACAP)

storeadmin mmp/imaproxy/popproxy/
 vdomain option, 27–27
 storeadminpass mmp/imaproxy/popproxy/
 vdomain option, 27–27
 RFC 2645 (On-demand Mail Relay), 33–132
 RFC 2798 (Definition of the inetOrgPerson
 LDAP Object Class), 39–102
 allow_unquoted_addrs_violate_rfc2798 MTA
 option, 39–92
 RFC 2852 (Deliver By SMTP service extension)
 deliverbymin channel option, 33–124
 RFC 3028 (Sieve: A Mail Filtering Language)
 Obsoleted by RFC 5228, 5–2
 See RFC 5228 (Sieve), 5–1
 RFC 3030 (SMTP Service Extensions for
 Transmission of Large and Binary MIME
 Messages), 33–117, 33–133
 RFC 3066 (Tags for the Identification of
 Languages), 47–9
 RFC 3280
 sslpkix base option, 7–13
 RFC 3339 (Date and Time on the Internet:
 Timestamps), 35–12
 ISO 8601 format, 1–3
 Value of LDAP attribute named by
 ldap_group_last_access_time, 39–134
 RFC 3461 (NOTARY)
 Section 5.2.7.1 mailing lists, Respond
 to delivery receipt requests during list
 expansion, 39–162
 RFC 3461 (SMTP DSN Extension), 33–96, 33–131
 RFC 3461–3464 (NOTARY), 47–1
 RFC 3598 (Sieve Email Filtering -- Subaddress
 Extension), 5–21, 5–43
 RFC 3749 (Transport Layer Security Protocol
 Compression Methods)
 sslcompress base option, 7–13
 RFC 3798 (Message Disposition Notification),
 47–1
 RFC 3834 (Recommendations for Automatic
 Responses to Electronic Mail)
 Auto-Submitted: header line on MDNs, 47–18
 Section 2. When (not) to send automatic
 responses, Address matching and the
 generation of vacation messages, 5–45
 RFC 3834 (Recommendations for Automatic
 Responses to Electronic MAIL), 5–46
 RFC 3865 (A No Soliciting SMTP Service
 Extension)
 destinationnosolicit and sourcenosolicit
 channel options, 33–124
 ldap_domain_attr_nosolicit MTA option,
 39–147
 ldap_nosolicit MTA option, 39–119
 RFC 3885 (SMTP Service Extension for Message
 Tracking), 48–1
 tracking* channel options, 33–91
 RFC 3886 (An Extensible Message Format for
 Message Tracking Responses), 48–1
 RFC 3887 (Message Tracking Query Protocol),
 33–91, 48–1
 RFC 3894 (Sieve Extension: Copying Without
 Side Effects), 5–22
 RFC 4314 (IMAP4 Access Control (ACL)
 Extension)
 capability_acl IMAP option, 21–5
 RFC 4314 (IMAP4 Access Control List (ACL)
 Extension), 33–42
 RFC 4408 (Sender Policy Framework)
 MTA options, 39–240
 Section 10.1. Processing Limits,
 spf_max_dns_queries MTA option, 39–244
 Section 10.1. Processing Limits, spf_max_time
 MTA option, 39–244
 RFC 4467 (IMAP - URLAUTH Extension), 49–7
 RFC 4468 (Message Submission BURL
 Extension), 49–7
 RFC 4511 (LDAP)
 Broken client authentication, 12–2
 RFC 4790 (Internet Application Protocol
 Collation Registry)
 Sieve comparators, 5–51
 RFC 4865 (SMTP Submission Service Extension
 for Future Message Release), 33–104, 33–126
 RFC 4954 (SMTP Service Extension for
 Authentication), 33–154
 RFC 5173 (Sieve Body Extension), 5–21
 RFC 5183 (Sieve Email Filtering: Environment
 Extension), 5–27
 RFC 5228 (Sieve: A Mail Filtering Language),
 5–1
 RFC 5228 (Sieve: An Email Filtering Language),
 5–2
 RFC 5229 (Sieve Email Filtering: Variables
 Extension), 5–46
 RFC 5230 (Sieve Email Filtering: Vacation
 Extension), 5–43
 RFC 5230 (Sieve Email Filtering: Vacation
 Extension)
 Section 4.2. Previous Response Tracking, 5–45
 Section 4.5. Address Parameter and Limiting
 Replies to Personal Messages, 5–45
 Section 4.6. Restricting Replies to Automated
 Processes and Mailing Lists, 5–45, 5–45, 44–15
 Section 4.7. Interaction with Other Sieve
 actions, 5–46

RFC 5231 (Sieve Email Filtering: Relational Extension), 5–17
 RFC 5232 (Sieve Email Filtering: Imap4flags Extension), 5–37
 RFC 5233 (Sieve: Subaddress Extension)
 subaddress* channel options, 33–42
 RFC 5235 (Sieve Email Filtering: Spamtest and Virustest Extensions), 5–42
 RFC 5256 (IMAP - SORT and THREAD Extensions)
 capability_sort IMAP option, 21–8
 RFC 5258 (IMAPv4 LIST Command Extensions), 21–6
 RFC 5260 (Sieve Email Filtering: Date and Index Extensions), 5–23
 RFC 5293 (Sieve Email Filtering: Editheader Extension), 5–25
 RFC 5322 (Internet Message Format)
 Message header and line containing solely white space, 33–73
 RFC 5429 (Sieve Email Filtering: Reject and Extended Reject Extensions), 5–28, 39–228
 RFC 5435 (Sieve Email Filtering: Extension for Notifications), 5–39
 RFC 5436 (Sieve Notification Mechanism: mailto), 5–39
 Auto-Submitted: header line on MDNs, 47–18
 RFC 5463 (Sieve Email Filtering: Ihave Extension), 5–37
 RFC 5703 (Sieve Email Filtering: MIME Part Tests, Iteration, Extraction, Replacement, and Enclosure), 5–38
 RFC 5746 (TLS Renegotiation Indication Extension)
 sslrequiresafenegotiate base option, 7–13
 RFC 5957 (Display-Based Address Sorting for the IMAP4 SORT Extension)
 capability_sort_display IMAP option, 21–8
 RFC 6009 (Sieve Email Filtering: Delivery Status Notifications and Deliver-By Extensions), 5–41
 envelope-dsn Sieve extension, 5–26
 RFC 6131 (Sieve Vacation Extension: "Seconds" Parameter), 5–43
 RFC 6134 (Sieve Extension: Externally Stored Lists), 5–29
 RFC 6154 (IMAP LIST Extension for Special-Use Mailboxes)
 capability_create_special_use IMAP option, 21–8
 capability_special_use IMAP option, 21–8
 RFC 6237 (IMAP4 Multimailbox SEARCH Extension)
 capability_multisearch IMAP option, 21–7
 RFC 6409 (Message Submission for Mail), 33–119
 RFC 6530 (Overview and Framework for Internationalized Email), G–4
 RFC 6710 (SMTP Extension for Message Transfer Priorities)
 Job Controller priority-based processing, 42–4
 mtppriorities* channel options, 33–106, 33–130
 RFC 6729 (Indicating Email Handling States in Trace Fields)
 receivedstate channel option, 33–77
 RFC 7293 (The Require-Recipient-Valid-Since Header Field and SMTP Service Extension), 33–35, 33–118
 RFC 822 (Internet Text Messages)
 Postmaster case insensitive local-part, 47–26
 RFC 976 (UUCP Mail Interchange Format Standard), 33–35
 RFC 976 (UUCP), 34–4
 Sieve
 draft-murchison-sieve-regex-08, 5–60
 RFC 3028, 5–2
 RFC 5228, 5–2
 Sieve refuse extension
 draft-elvey-refuse-sieve-01.text, 5–28
 URL
 See Standards, RFC 1738 (Uniform Resource Locators (URL)), 35–42
 statinterval alarm.system:diskavail option, 11–2
 statinterval alarm.system:serverresponse option, 11–3
 storage local_table MeterMaid option, 46–3
 Store options
 See Message Store options, 16–3
 storeadmin mmp/imaproxy/popproxy/vdomain option, 27–27
 storeadminpass mmp/imaproxy/popproxy/vdomain option, 27–27
 storehostlist proxy option, 26–2
 streaming channel option, 33–134
 stressblackout Job Controller option, 42–14
 stressfactor Job Controller option, 42–15
 stressfdwait base option, 7–5
 stressjobs Job Controller option, 42–15
 stresslimit Message Store checkpoint option, 16–16
 stressperiod base option, 7–5
 stresstime Job Controller option, 42–14
 strict_require MTA option
 Sieve extensions, 5–18
 string_pool_size_3 MTA option
 General database, 37–21
 style Message Store archive option, 16–14
 Subaddresses
 Alias match, 35–45

- alias_domains MTA option, 39–56
- deliveryflags channel option, 33–108, 33–123
- fileinto channel option, 33–110
- ims-ms channel interpretation of
 - FILEINTO ims-ms-channel-specific option, 51–6
- In aliases, 35–44
- Mailing list members, 36–22
- Meta-groups, 39–100
- Sieve subaddress extension, 5–43
- subaddress* channel options, 33–42
- subaddress_char MTA option, 39–60
- subaddressexact channel option, 33–42
 - Address reversal, 35–55
- subaddressrelaxed channel option, 33–42
 - Address reversal, 35–55
 - Subaddresses on aliases, 35–45
- subaddresswild channel option, 33–42
 - Address reversal, 35–55
- subaddress_char MTA option, 39–60
- subdirs channel option, 33–59
- submit channel option, 33–119
- SUBMIT Proxy
 - Options, 27–3
 - banner, 27–7
 - connlimits, 27–10
 - debugkeys, 27–11
 - defaultdomain, 27–12
 - domainallowed, 27–13
 - domainnotallowed, 27–13
 - ehlokeywords, 27–14
 - ehlokeywords, Analogous to capability IMAP proxy option, 27–9
 - failovertimeout, 27–15
 - ldapcachesize, 27–16
 - ldapcachettl, 27–16
 - ldapurl, DEPRECATED, 27–17
 - loglevel, 27–18, 39–72
 - popbeforesmtpkludgechannel, 27–20
 - preauthtimeout, 27–20
 - smtprelays, 27–23
 - ssladjustciphersuites, 7–11, 27–24
 - tcpaccess, 27–28
 - timeout, 27–28
 - usergroupdn, DEPRECATED; see ugldapbasedn instead, 27–29
 - use_nslog, 27–29
 - virtualdomainfile, DELETED; see vdomain options instead, 27–29
- submituser IMAP option, 21–4, 39–68
 - BURL usage, 49–11
- subscribesynclevel Message Store option, 16–9
- substring_search indexer option, 25–2
- suffix_search indexer option, 25–2
- supportedlanguages base option, 7–13
- suppressfinal channel option, 33–95
 - Recipient address reported in notifications, 47–6, 47–17
- switchchannel channel option, 33–81
 - CHECK_SOURCE TCP/IP-channel-specific option, 49–22
 - Effectively disabled if CHECK_SOURCE=0, 49–22
 - INTERNAL_IP mapping table, 44–6
- Symantec Corp.
 - Certificate Authority, G–1
- synch_time Job Controller option, 42–15
 - Operation, 42–3
- syncdap IMAP Proxy and POP Proxy option, 27–27
- synclevel Message Store option, 16–9
- Syslog notices
 - \$< flag in PORT_ACCESS mapping table, 44–4
 - \$> flag in PORT_ACCESS mapping table, 44–4
 - .HELD messages, 39–220, 39–246
 - From ims-ms channels, 51–6
 - Generated by address access mapping tables, 44–9
 - HELDMSG, Header count exceeded; message has been marked .HELD automatically
 - Diagnosing .HELD files, 52–11
 - HELDMSG, Header count exceeded; message has been marked .HELD automatically., 39–220, 39–246
 - HELDMSG, Max MIME part/level limit exceeded; message has been marked .HELD automatically., 39–221, 39–246
 - HELDMSG, Max recipient count exceeded; message has been marked .HELD automatically., 39–221, 39–246
 - MTA options, 39–246
 - MTA transaction entries
 - Line length maximum, 39–247, 39–249
 - PORT_ACCESS mapping table, 44–3
 - SPAMFILTERn, Cannot find DIRECTORY value in options, 45–9
 - SPAMFILTERn, Cannot find style value in options, 45–9
 - SPAMFILTERn, Error opening archive options file, 45–9
 - SPAMFILTERn, Error reading archive options file, 45–9
 - SPAMFILTERn, Error reading ClamAV options file, 45–4
 - SPAMFILTERn, Error reading ICAP options file, 45–5

- SPAMFILTERn, Error reading Milter options file, 45–5
- SPAMFILTERn, Error reading SpamAssassin options file, 45–7
- SPAMFILTERn, No host specified in ClamAV option file, 45–4
- SPAMFILTERn, No host specified in SpamAssassin option file, 45–7
- TLS_ACCESS mapping table, 49–46
- syslogfacility logfile option, 7–21
- syslogfacility option
 - ims-ms channels, 51–6
- systemfilter MTA option, 39–223, 47–2
 - Performance impact, 56–3
 - Sieve hierarchy, 5–65
- system_id SMS smpp_server option, 53–13

T

- table_options local_table MeterMaid option, 46–3
- table_type local_table MeterMaid option, 46–3
- Task options
 - See Scheduler task options, 8–1
- TCP wrappers, 6–1
 - dnsresolveclient option, 7–16
 - domainallowed option, 27–13
 - domainnotallowed option, 27–13
 - Examples, 6–5
 - Restricting access to require use of POPS and IMAPS, 6–3
 - Filter creation, 6–7
 - Filter syntax, 6–2
 - EXCEPT operator, 6–5
 - http vs. mshttpd service-name, altservice
 - MSHTTP option, 29–11
 - server-host specification, 6–5
 - Wildcard names, 6–4
 - Wildcard patterns, 6–4
 - tcpaccess MMP/IMAP Proxy/POP Proxy/SUBMIT Proxy/vdomain option, 27–28
- TCP/IP channels, 49–2
 - Authentication errors, 49–52
 - AUTH_ACCESS mapping table, 49–37
 - Connection history caching, 33–135
 - Debug log files, 33–85
 - Dial up connections
 - SMTP ETRN extension, 49–51
 - DNS lookups
 - Channel options, 33–135
 - Incoming connections, forwardcheck* channel options, 33–139
 - Incoming connections, ident* channel options, 33–138
 - Outgoing connections, 33–137

- Examples, 49–4
- Fast disconnect flag
 - SO_LINGER, \$/ flag in address *_ACCESS mapping tables, 44–9
 - SO_LINGER, \$/ flag in PORT_ACCESS mapping table, 44–4
- Local host table name lookups, 33–137
- MX records, 33–137
- Nameserver selection, 33–137
- Options, 49–16
 - 552_PERMANENT_ERROR_STRING, 49–17
 - ALLOW_ETRNS_PER_SESSION, 49–18
 - ALLOW_RECIPIENTS_PER_TRANSACTION, 49–18, 49–18
 - ALLOW_RECIPIENTS_PER_TRANSACTION, Compared to channel options, 33–87, 33–121
 - ALLOW_REJECTIONS_BEFORE_DEFERRAL, 49–19
 - ALLOW_REJECTIONS_BEFORE_DEFERRAL, Compared to channel options, 33–87, 33–121
 - ALLOW_SESSION_BLOCKS, 49–19
 - ALLOW_TRANSACTIONS_PER_SESSION, 49–20
 - ALLOW_TRANSACTION_BLOCKS, 49–19
 - ATTEMPT_TRANSACTIONS_PER_SESSION, 49–20
 - ATTEMPT_TRANSACTIONS_PER_SESSION, BSMTP channels, 50–3
 - AUTH_DEBUG, 49–20
 - AUTH_DEBUG, \$A flag in PORT_ACCESS mapping table, 44–4
 - AUTH_PASSWORD, 49–20
 - AUTH_PASSWORD, *saslient channel options, 33–150
 - AUTH_USERNAME, 49–20
 - AUTH_USERNAME, *saslient channel options, 33–150
 - BANNER_ADDITION, 49–21
 - BANNER_HOST, 33–80, 49–21
 - BANNER_HOST, Local channel official_host_name, 52–2
 - BANNER_PURGE_DELAY, 49–21
 - BANNER_PURGE_DELAY, \$D flag in PORT_ACCESS mapping table, 44–4
 - BANNER_PURGE_DELAY, Example setting, 49–16
 - BANNER_REVERSE_HOST, 49–21
 - BUFFER_SIZE, 39–173, 49–22
 - CHECK_SOURCE, 49–22
 - CLIENT_CERT_NICKNAME, 49–22
 - COMMAND_RECEIVE_TIME, 49–22
 - COMMAND_TRANSMIT_TIME, 49–23
 - CONTINUATION_CHARS, 49–23

CUSTOM_VERSION_STRING, 49-23
 DATA_RECEIVE_TIME, 49-23
 DATA_TRANSMIT_TIME, 49-23
 DISABLE_ADDRESS, 49-23
 DISABLE_ADDRESS, \$V flag in
 PORT_ACCESS mapping table, 44-4
 DISABLE_CIRCUIT, 49-24
 DISABLE_CIRCUIT, \$V flag in
 PORT_ACCESS mapping table, 44-4
 DISABLE_EXPAND, 49-24
 DISABLE_EXPAND, expr* channel options,
 33-126
 DISABLE_GENERAL, 49-25
 DISABLE_GENERAL, \$V flag in
 PORT_ACCESS mapping table, 44-4
 DISABLE_SEND, 49-25
 DISABLE_STATUS, 49-25
 DISABLE_STATUS, \$V flag in
 PORT_ACCESS mapping table, 44-4
 DOT_TRANSMIT_TIME, 49-26
 EXTERNAL_IDENTITY, 49-20
 EXTERNAL_IDENTITY, *saslient channel
 options, 33-150
 FAST_SMTP_SESSION_TIME_LIMIT, 49-26
 HELLO_RECEIVE_TIME, 49-26
 HIDE_VERIFY, 49-26
 IGNORE_BAD_CERT, 49-27
 INITIAL_COMMAND, 49-27
 LOG_BANNER, 49-27
 LOG_BANNER, MTA logging, 39-251
 LOG_CONNECTION, 49-27
 LOG_CONNECTION, MTA logging, 39-251
 LOG_TRANSPORTINFO, 49-28
 LOG_TRANSPORTINFO, MTA logging,
 39-251
 MAILBOX_BUSY_FAST_RETRY, 49-29
 MAIL_TRANSMIT_TIME, 49-28
 MAX_A_RECORDS, 49-29
 MAX_B_ENTRIES, 49-29
 MAX_B_ENTRIES, MTA logging, 39-251
 MAX_CLIENT_THREADS, 49-29
 MAX_HELO_DOMAIN_LENGTH, 49-30
 MAX_J_ENTRIES, 49-30
 MAX_J_ENTRIES, Diagnostic text in MTA
 transaction log entries, 55-20
 MAX_J_ENTRIES, MTA logging, 39-251
 MAX_MX_RECORDS, 49-30
 MAX_SERVER_THREADS, 49-31
 OPEN_CONNECTION_TIME, 49-31
 PACKET_SIZE_LIMIT, 49-31
 PROXY_PASSWORD, 49-31
 RCPT_TRANSMIT_TIME, 49-32
 RECIPIENT_DELAY_AMOUNTS, 49-34
 RECIPIENT_DELAY_THRESHHOLDS, 49-34
 REJECT_RECIPIENTS_PER_TRANSACTION,
 49-32
 REJECT_RECIPIENTS_PER_TRANSACTION,
 Compared to channel options, 33-87, 33-121
 REUSE_TIMED_OUT_TRANSFERS, 49-32
 SESSION_TIME, 49-33
 SIZE_DELAY_AMOUNTS, 49-34
 SIZE_DELAY_THRESHHOLDS, 49-34
 SSL_CLIENT, 49-34
 SSL_CLIENT, AUTH_ACCESS mapping table
 \$D flag, 49-38
 STARTTLS_FAILURE_RECONNECT_DELAY,
 49-35
 STATUS_DATA_RECEIVE_TIME, 49-35
 STATUS_DATA_RECV_
 PER_ADDR_PER_BLK_TIME, 49-35
 STATUS_DATA_RECV_PER_ADDR_TIME,
 49-35
 STATUS_DATA_RECV_PER_BLOCK_TIME,
 49-35
 STATUS_MAIL_RECEIVE_TIME, 49-36
 STATUS_RCPT_RECEIVE_TIME, 49-36
 STATUS_RECEIVE_TIME, 49-36
 STATUS_TRANSMIT_TIME, 49-36
 TRACE_LEVEL, 49-36
 TRANSACTION_DELAY_AMOUNTS, 49-34
 TRANSACTION_DELAY_THRESHHOLDS,
 49-34
 TRANSACTION_LIMIT_RCPT_TO, 49-36
 TRANSACTION_TIME, 49-37
 WINDDOWN_TIMEOUT, 49-37
 Performance
 MAX_CLIENT_THREADS TCP/IP-channel-
 specific option, 49-30
 Routing
 Gateway systems, 49-46
 Mailhubs, 49-47
 Typical basic configuration, 49-4
 tcpaccess MMP/IMAP Proxy/POP Proxy/SUBMIT
 Proxy/vdomain option, 6-7, 27-28
 tcpaccessattr MMP option
 Alternate LDAP attribute in place of
 mailAllowedServiceAccess, 39-103
 tcpaccessattr MMP/IMAP Proxy/POP Proxy option,
 6-7
 tcpaccessattr MMP/IMAP Proxy/POP Proxy/
 vdomain option, 27-28
 tcpaccessattr option, 6-8
 TCP wrapper access filters, 6-2
 tcp_listen options, 28-1
 backlog, 28-1
 listen_addresses, 28-1

- ssl_ports, 28-1
- tcp_ports, 28-1, 41-10
- tcp_lmtp_server options
 - See LMTP channels, Server, Options, 49-15
- tcp_ports Dispatcher option, 41-9
- tcp_ports Job Controller option, 41-10, 42-15
- tcp_ports SMS smpp_relay option, 41-9, 41-9, 53-10
- tcp_ports SMS smpp_server option, 41-9, 41-10, 53-13
- tcp_ports tcp_listen option, 28-1, 41-10
- Text attachments
 - Character set, 38-16
- Text messaging
 - SMS, 53-2
- text_to_subject gateway_profile option, 53-4
 - email_body_charset gets ignored, 53-5
- Thawte Consulting
 - Certificate Authority, G-1
- Third party submission
 - AUTH_ACCESS mapping, 49-39
- thirdclassafter channel option, 33-100
- threaddepth channel option, 33-106, 33-146
 - Channel to a gateway system, 49-47
 - Job Controller operation, 42-2
- thresholddelay base option, 7-13
- thread_count_initial sms_gateway option, 53-4
- thread_count_maximum sms_gateway option, 53-4
- thread_stack_size sms_gateway option, 53-4
- threshold alarm.system:diskavail option, 11-3
- threshold alarm.system:serverresponse option, 11-3
- thresholddirection alarm.system:diskavail option, 11-3
- thresholddirection alarm.system:serverresponse option, 11-3
- thurman channel option, 33-50
- tick channel option, 33-52
- timeout Autorestart option, 7-23
- timeout IMAP Proxy option, 27-28
- timeout indexer option, 25-2
- timeout MeterMaid Client option, 46-5
- timeout MMP option, 27-28
- timeout msprobe option, 10-1
- timeout POP Proxy option, 27-28
- timeout SUBMIT Proxy option, 27-28
- Timeouts
 - Archived message retrieval
 - retrievetimeout Message Store archive option, 16-15
 - ClamAV responses
 - TIMEOUT ClamAV config file option, 45-4
 - Direct LDAP lookups
 - Caching, 39-152
 - Domain map cache
 - ldap_timeout MTA option, 7-5, 39-83, 39-153
 - ICAP responses
 - TIMEOUT ICAP option, 45-5
 - Incoming SMTP sessions
 - COMMAND_RECEIVE_TIME TCP/IP-channel-specific option, 49-22
 - DATA_RECEIVE_TIME TCP/IP-channel-specific option, 49-23
 - Expansion of multiple recipient address, 52-18
 - HELLO_RECEIVE_TIME TCP/IP-channel-specific option, 49-26
 - REUSE_TIMED_OUT_TRANSFERS TCP/IP-channel-specific option, 49-33
 - SESSION_TIME TCP/IP-channel-specific option, 49-33
 - STATUS_TRANSMIT_TIME TCP/IP-channel-specific option, 49-36
 - TRANSACTION_TIME TCP/IP-channel-specific option, 49-37
 - LDAP modify operations
 - ldapmodifytimeout base option, 7-10
 - LDAP queries
 - ADMLDAP_TIMEOUT environment variable, 39-77
 - Caching, 39-152
 - ldapssearchtimeout base option, 7-10, 39-77
 - ldaptimeout mmp/imaproxy/popproxy option (DEPRECATED), 27-17
 - ldap_timeout MTA option, 39-77
 - local.ldapssearchtimeout configutil parameter, 39-77
 - Milter connections
 - CONNECT_TIMEOUT Milter option, 45-5
 - Milter responses
 - TIMEOUT Milter option, 45-5
 - MTQP connections
 - mtqp_timeout MTA option, 39-212
 - Outgoing SMTP sessions
 - COMMAND_TRANSMIT_TIME TCP/IP-channel-specific option, 49-23
 - DATA_TRANSMIT_TIME TCP/IP-channel-specific option, 49-23
 - DOT_TRANSMIT_TIME TCP/IP-channel-specific option, 49-26
 - MAIL_TRANSMIT_TIME TCP/IP-channel-specific option, 49-28
 - RCPT_TRANSMIT_TIME TCP/IP-channel-specific option, 49-32
 - STATUS_DATA_RECEIVE_TIME TCP/IP-channel-specific option, 49-35

- STATUS_DATA_RECV_PER_ADDR_PER_BLK_TIME TCP/IP-channel-specific option, 49–35
- STATUS_DATA_RECV_PER_ADDR_TIME TCP/IP-channel-specific option, 49–35
- STATUS_DATA_RECV_PER_BLOCK_TIME TCP/IP-channel-specific option, 49–35
- STATUS_MAIL_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
- STATUS_RCPT_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
- STATUS_RECEIVE_TIME TCP/IP-channel-specific option, 49–36
- WINDDOWN_TIMEOUT TCP/IP-channel-specific option, 49–37
- POP before SMTP
 - authservicetl POP Proxy/vdomain option, 27–6
- Reversal cache
 - reverse_address_cache_timeout MTA option, 39–153
- Sieve notify messages
 - notify_maximum_timeout MTA option, 39–65
 - notify_minimum_timeout MTA option, 39–66
- SMTP server connection
 - MMP, failovertimeout option, 27–15
- Spam/virus filter package integration
 - Milter, CONNECT_TIMEOUT Milter option, 45–5
 - Milter, TIMEOUT Milter option, 45–5
 - SpamAssassin, CONNECT_TIMEOUT SpamAssassin option, 45–7
 - SpamAssassin, TIMEOUT SpamAssassin option, 45–7
- Spam/virus filter package responses
 - ClamAV, TIMEOUT ClamAV config file option, 45–4
 - ICAP, 45–5
- SpamAssassin connections
 - CONNECT_TIMEOUT SpamAssassin option, 45–7
- SpamAssassin responses
 - TIMEOUT SpamAssassin option, 45–7
- User authentication cache
 - authcachettl option, 7–3, 27–5, 27–6
- User entry cache (MMP)
 - ldapcachettl option, 27–16
- Vacation messages
 - ldap_autoreply_timeout MTA option, 39–128
 - ldap_domain_attr_autoreply_timeout MTA option, 39–147
 - vacation_maximum_timeout MTA option, 39–66, 39–102
 - vacation_minimum_timeout MTA option, 39–66, 39–102
- timestampdelta smime option, 30–5
- TLS
 - tlsused switch of test -rewrite utility, 58–38
 - Information included in certain *_access mapping table probes, 44–8
 - MTA options
 - sslnicknames, 27–27, 39–218
 - S modifier in MTA message transaction log entries, 55–4
 - SMTP
 - Required for implicitsaslexternal to take effect, 33–151
 - Testing for use in *_ACCESS mapping table entries, 44–12
 - tls_bits_reject_msg Dispatcher service option, 41–10
 - tls_min_bits Dispatcher service option, 41–10
 - tlsswitchchannel channel option, 33–83, 33–151
 - tlsv12enable base option, 7–18
 - tls_bits_reject_msg Dispatcher option, 41–10
 - tls_min_bits Dispatcher option, 41–10
 - tmpdir base option, 7–14
 - tmpdir Message Store archive option, 16–14
 - tmpdir MTA option, 39–154
 - Effect on location of MTA database temp files, 40–4
 - tokenpass sectoken option, 13–1
- Top Level Domains (TLDs)
 - error_text_unknown_host MTA option, 39–158
 - Informing MTA of updated list, 58–9
 - Rewrite rule check, 34–32
 - tlds.txt file, 34–10, 34–32
- tracking* channel options, 33–91
- trackinggenerate channel option, 33–92, 33–92
- tracking_debug MTA option, 39–75
- tracking_mode MTA option, 39–210
- tracking_retries MTA option, 39–210
- tracking_retry_delay MTA option, 39–210
- transactionlimit channel option, 33–124
 - error_text_transaction_limit_exceeded MTA option, 39–166
 - Reprocess channel, 52–19
- Transport information
 - include_connectioninfo MTA option, 39–189
 - log_connection MTA option, 39–254
 - LOG_CONNECTION TCP/IP-channel-specific option, 49–27
 - PORT_ACCESS mapping table, 44–3
 - Reprocess channel *_ACCESS mapping table probes, 52–19
 - Rewrite rule template substitutions, 34–26

- Syntax of, 55–7
 - TRANSPORTINFO environment variable
 - test_smtp_master and test_smtp_slave use of, 52–9
 - Troubleshooting
 - .HELD files, 52–11
 - DSNs that are "incomplete", 47–9
 - truncatesmtpplonglines channel option, 33–133
 - trustedurl smime option, 30–2
 - tspecials (from RFC 2045), 33–77
 - tty notifytarget option, 24–4
 - Tunnelling messages between MTAs, 50–1
 - turn channel option, 33–132
 - turn_in channel option, 33–132
 - turn_out channel option, 33–132
- ## U
- ugldapbasedn base option, 7–14
 - Direct LDAP alias lookups, 35–6
 - ugldapbindcred base option, 7–14
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - ugldapbinddn base option, 7–14
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - ugldaphost base option, 7–14
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - ugldapport base option, 7–14
 - Direct LDAP alias lookups, 35–5
 - Direct LDAP domain lookups, 34–29
 - ugldapusessl base option, 7–15
 - Direct LDAP alias lookups, 35–5
 - umask Message Store option, 16–13
 - underquota notifytarget option, 24–5
 - Unified Configuration, 1
 - unique_id_template MTA option
 - Message identifier generation (for archiving), 54–19
 - unrestricted channel option, 33–40
 - unstressfactor Job Controller option, 42–15
 - unstressjobs Job Controller option, 42–15
 - updatemsg notifytarget option, 24–5
 - urgentafter channel option, 33–100
 - urgentbackoff channel option, 33–101
 - urgentblocklimit channel option, 33–114
 - Job Controller delivery execution window, 42–16
 - urgentnotices channel option, 33–96
 - urgent_block_limit MTA option, 39–210, 39–220
 - urgent_delivery Job Controller job_pool option, 42–16
 - url_result_cache_size MTA option, 39–154
 - usedomainmap auth option, 12–1
 - useheaderrecipients Message Store archive option, 16–15
 - useintermediate channel option, 33–95
 - Recipient address reported in notifications, 47–6, 47–17
 - usepermanenterror channel option, 33–57
 - user channel option, 33–62, 33–107
 - Pipe channels, 52–14
 - See also pipeuser option in restricted.cnf, 33–62, 33–107
 - user Dispatcher option, 41–10
 - user Dispatcher service option, 41–10
 - user Message Store publicsharedfolders option, 16–25
 - usercertfilter smime option, 30–1
 - usereplyto channel option, 33–78
 - useresent channel option, 33–78
 - usergroupdn MMP/IMAP Proxy/POP Proxy/SUBMIT Proxy option
 - DEPRECATED: see ugldapbasedn instead, 27–29
 - userid Deployment Map option, 14–2
 - Users
 - Duplicate/ambiguous LDAP entries
 - error_text_duplicate_addr MTA option, 39–162
 - Forwarding, 35–57
 - ldap_forwarding_address MTA option, 39–130
 - Techniques, 35–59
 - Head-of-household controls
 - ldap_filter_reference MTA option, 39–129
 - ldap_parental_controls MTA option, 39–129
 - LDAP attributes
 - Auto-secretary use, 39–122
 - Capture trigger, 39–116
 - expandable, 39–140
 - inetUserStatus, 39–114
 - mail, 39–120
 - mail, Fetched during head of household Sieve filter lookups, 39–130
 - mailAlternateAddress, 39–121
 - mailConversionTag, 39–123
 - mailDeliveryFile, 39–125
 - mailDeliveryFileURL, 39–125
 - mailDeliveryOption, 39–120
 - mailDeliveryOption, Forwarding mail, 35–59
 - mailEquivalentAddress, 39–121
 - mailForwardingAddress, Forwarding mail, 35–59
 - mailHost, 39–124
 - mailHost,
 - ldap_domain_attr_default_mailhost MTA option, 39–147

- mailMsgMaxBlocks, 39–123
- mailMsgQuota, 39–125
- mailProgramDeliveryInfo, 39–125
- mailQuota, 39–124
- mailRoutingAddress, 39–119
- mailSieveRuleSource, Fetched during head of household Sieve filter lookups, 39–130
- mailUserStatus, 39–114
- mgmanMemberVisibility, 39–140
- MLS range, 39–116
- NO-SOLICITING values, 39–119
- objectClass, 39–114
- Opt-in to detour routing, 39–123
- Opt-in to spam package N, 39–121
- Personal name, 39–120
- Preferred country labelling, 39–119
- preferredLanguage, 39–118
- Presence flag, 39–122
- Recipient cutoff, 39–117
- Recipient limit, 39–117
- rfc822mailalias, 39–121
- Source block limit, 39–118
- Source channel switch, 39–118
- Source conversion tag, 39–120
- Source opt-in to spam package N, 39–118
- Spare N attribute, 39–125
- uid, 39–115
- vacationEndDate, 39–123
- vacationStartDate, 39–122
- Parental controls
 - ldap_filter_reference MTA option, 39–129
 - ldap_parental_controls MTA option, 39–129
- Passwords
 - Expiration warnings, 21–13
 - Plaintext, Forbidden unless SSL/TLS is active, 27–21
 - Plaintext, has_plain_passwords Auth option, 12–1
 - userPassword LDAP attribute, 39–103
- Quota
 - defaultmailboxquota Message Store option, 16–11
 - defaultmessagequota Message Store option, 16–11
 - ldap_disk_quota MTA option, 39–124
 - ldap_message_quota MTA option, 39–125
 - Over quota status, Customizing MTA error text, 39–157
 - Over quota status, error_text_over_quota MTA option, 39–161
 - Over quota status, IMAP ALERT message, 47–3
 - Over quota status, Notification message from Message Store, 47–3
 - Over quota status, overquotastatus Message Store option, 16–7
 - Over quota status, quotaexceeded Message Store option, 16–13
 - Over quota status, quotaexceededmsginterval Message Store option, 16–12
 - Over quota status, quotagraceperiod Message Store option, 16–12
 - Over quota status, quotanotification Message Store option, 16–12
 - Over quota status, Reported to entire group membership, 36–17
 - Over quota status, SMTP rejection text, 39–161
 - quotaenforcement Message Store option, 16–12
 - quotaoverdraft Message Store option, 16–8
 - See also Message Store folderquota options, 16–20
 - See also Message Store msgagetype and typequota options, 16–20
 - subdirs channel option on ims-ms channel, 51–1
 - Warning that user is nearing limit, quotawarn Message Store option, 16–13
- Sieve filters
 - ldap_filter MTA option, 39–129
 - ldap_filter_reference MTA option, 39–129
- Testing of LDAP entries
 - test -rewrite utility, 58–28
- Vacation
 - ldap_autoreply_addresses MTA option, 39–128
 - ldap_autoreply_mode MTA option, 39–126
 - ldap_autoreply_subject MTA option, 39–126
 - ldap_autoreply_text MTA option, 39–127
 - ldap_autoreply_text_internal MTA option, 39–128
 - ldap_autoreply_timeout MTA option, 39–128
 - vnd.sun.autoreply-internal envelope item in Sieve filters, 5–27
- userswitchchannel channel option, 33–81
 - Address reversal, 35–50
- usesentdate MSHTTP option, 29–6
- usetemporaryerror channel option, 33–57
- use_alias_database MTA option, 39–60
- use_auth_return MTA option, 39–194
 - Effect on FORWARD mapping table probe, 35–59
 - Effect on use_moderator_url attribute's value, 39–134

- use_canonical_return MTA option, 39–194
 - Effect on FORWARD mapping table probe, 35–59
 - Effect on use_moderator_url attribute's value, 39–134
- use_comment_strings MTA option, 39–198
- use_domain_database MTA option, 39–61
- use_forward_database MTA option, 39–61, 39–199
 - Effect on FORWARD mapping table probe, 35–59, 35–59
- use_ip_access MTA option, 39–194
- use_mail_delivery MTA option, 39–273
- use_nslog Dispatcher option, 27–29, 41–10
- use_nslog Job Controller option, 27–29, 42–16
- use_nslog option, 27–29
- use_orig_return MTA option, 39–194
 - Effect on FORWARD mapping table probe, 35–59
 - Effect on use_moderator_url attribute's value, 39–134
- use_permanent_error MTA option, 39–168
 - recipientlimit channel option, 33–87, 33–121
- use_personal_names MTA option, 39–201
- use_precedence MTA option, 39–217
- use_reverse_database MTA option, 39–62, 39–200
 - Limiting emission of internal host names, 57–3
 - Message-id: modification, 57–3
- use_sms_priority SMS gateway option, 53–8
- use_sms_privacy SMS gateway option, 53–9
- use_temporary_error MTA option, 39–169
- use_text_databases MTA option, 39–175
 - Reverse database, 35–52
 - Rewrite rule general database substitutions, 34–24
- use_warnings_to MTA option, 39–217
- utf8strict channel option
 - error_text_unnegotiated_eightbit MTA option, 39–166
- Utilities, 58–2
 - cache -change, 58–3
 - Message replay example, 54–5
 - cache -sync, 58–6
 - synch_time Job Controller option, 42–15
 - cache -synchronize
 - Message replay example, 54–5
 - cache -walk, 58–8
 - chbuild, 58–9
 - charset options file, option_charset.dat, 40–9
 - imta_charset_data MTA tailor option, 40–6
 - clbuild
 - imta_command_data MTA tailor option, 40–6
 - cnbuild, 58–12
 - imta_config_data MTA tailor option, 40–6
- counters -show
 - Example of Sieve counters, 5–50
- crdb, 58–19
 - Use with the alias database, 35–44
- imarchive, 54–20
- imcheck
 - Message Store cache efficiency, 16–10
- imexpire
 - Archiving, 54–20
 - channel, 45–17
 - IMAP user flags, 5–38
 - rescanhours, 45–17
 - Sieve body extension, 5–22
 - Spam/virus filter package integration, 45–2, 45–16
 - Spam/virus filter packages, sourcespamfilter* and sourcespamfilter*optin channel options, 33–115
 - Spamfilter integration, scan_channel MTA option, 39–170
 - Spamfilter integration, scan_originator MTA option, 39–170
 - Spamfilter integration, scan_recipient MTA option, 39–170
- ims_svc_*
 - softtokendir base option, 7–17
- msconfig
 - EDIT CONVERSIONS, 39–69
 - EDIT MAPPINGS, 37–3
 - EDIT REWRITES, 34–2
 - Macro substitutions, 39–215
 - Recipe language, 4–1
 - Used to configure Messaging Server, 1
- process, 58–22
- profile
 - Example, 52–15
- purge, 58–23
- qm
 - alias_username alias option, 35–23
 - Commands, release and the Hold channel, 52–11
 - Commands, unstress, 42–4
 - start, Message replay example, 54–5
 - stop, hold_list file, 40–9
 - stop, Message replay example, 54–5
 - stress, 42–4
 - USERNAME alias file named parameter, 35–40
- reload, 58–25
- return
 - return_bounced.txt, 47–12
- test -expression
 - Sieve external lists, 5–36

- test -mapping
 - Testing of mapping tables, 37–22
- test -match, 58–26
 - Testing mapping table patterns, Wildcards, 37–5
 - Testing of mapping tables, 37–22
- test -mime
 - Content-transfer-encoding: testing, 33–47
- test -rewrite, 58–28
 - Access control testing, 58–30
 - Caret quoting of input, 58–30
 - DNS verification, 58–35
 - Input characters by ASCII code, 58–30
 - Testing address access mapping tables, 37–23
 - Testing of mapping tables, 37–23
- uwcontexturi MSHTTP option, 29–9
- uwcenabled MSHTTP option, 29–9
- uwchome MSHTTP option, 29–10
- uwclogouturl MSHTTP option, 29–10
- uwcport MSHTTP option, 29–9
- uwcsslport MSHTTP option, 29–9

V

Vacation messages

Format of

- ldap_autoreply_mode MTA option, 39–126
- mailAutoReplyMode LDAP attribute, 39–126
- See Sieve filters, vacation action, :echo or :reply, 5–43

MTA options

- ldap_autoreply_addresses, 39–128
- ldap_autoreply_mode, 39–126
- ldap_autoreply_subject, 39–126
- ldap_autoreply_text, 39–127
- ldap_autoreply_text_internal, 39–128, 39–128
- ldap_autoreply_timeout, 39–128
- ldap_domain_attr_autoreply_timeout, 39–147
- ldap_end_date, 39–123
- ldap_start_date, 39–122
- max_vacations, 39–227
- vacation_cleanup, 39–66
- vacation_maximum_timeout, 39–66, 39–102
- vacation_minimum_timeout, 39–66, 39–102
- vacation_template, 39–67

Own addresses recognized

- ldap_autoreply_addresses MTA option, 39–128
- See Sieve filters, vacation action, :addresses or :noaddresses, 5–43

Previous response database

- Error accessing means vacation message not generated, 5–45
- MTA options, 39–64

Repeat of

- autoreply_timeout_default MTA option, 39–65
- ldap_autoreply_timeout MTA option, 39–128
- ldap_domain_attr_autoreply_timeout MTA option, 39–147
- mailAutoReplyTimeout LDAP attribute, 39–128
- See Sieve filters, vacation action, :days or :hours or :seconds, 5–43
- vacation_maximum_timeout MTA option, 39–66, 39–102
- vacation_minimum_timeout MTA option, 39–66, 39–102
- vacation_template MTA option, 39–67

Sieve filter vacation extension, 5–43

Subject of

- ldap_autoreply_subject MTA option, 39–126
- mailAutoReplySubject LDAP attribute, 39–126
- See Sieve filters, vacation action, :subject, 5–43

Text of

- ldap_autoreply_text MTA option, 39–127
- ldap_autoreply_text_internal MTA option, 39–128
- mailAutoReplyText LDAP attribute, 39–127
- mailAutoReplyTextInternal LDAP attribute, 39–128
- See Sieve filters, vacation action, 5–43

Time range

- ldap_end_date MTA option, 39–123
- ldap_start_date MTA option, 39–122
- See Sieve filters, date test, 5–43
- vacationEndDate LDAP attribute, 39–123
- vacationStartDate LDAP attribute, 39–122
- vacation_maximum_timeout MTA option, 39–66, 39–102
- vacation_minimum_timeout MTA option, 39–66, 39–102

Why not generated, 5–45

- Domain not properly defined in LDAP or not found, 5–45
- Domain, user or group status, 5–45
- Incompatible other Sieve action performed, 5–46
- mailAutoreplyText LDAP attribute or value missing, 5–46
- Original message a list post per header lines, 5–45
- Original message disabled all notifications, 5–45
- Original message's From address suggests list post, 5–45

- Outside vacationStartDate-vacationEndDate range, 5–45
 - Recipient address not present in original message header, 5–45
 - Recipient not properly defined in LDAP or not found, 5–45
 - Same vacation response already sent recently, 5–45
 - Sieve novacation or FROM_ACCESS \$! applied, 5–45
 - Sieve vacation action syntax error, 5–46
 - Too many vacation actions already performed in Sieve script, 5–45
 - Trouble accessing vacation-previous-response database, 5–45
 - Vacation action not supported in system Sieves, 5–46
 - vacation_minimum_timeout MTA option
 - Vacation message not generated, 5–45
 - value_type local_table MeterMaid option, 46–3
 - Vanity domains, G–11
 - Direct LDAP lookups
 - domain_match_url MTA option, 34–30
 - domain_match_url MTA option, 39–80
 - Example, 35–8
 - Venema, Wietse
 - Unix Tcpsd access-control, 6–1
 - verb_never channel option, 33–52
 - verb_none channel option, 33–52
 - verb_off channel option, 33–52
 - verb_on channel option, 33–52
 - verifcert Base certmap option, 7–23
 - verifcert certmap option, 7–23
 - verifyurl SSO option, 31–1
 - Verisign, Inc.
 - Certificate Authority, G–1
 - VERP (Variable Envelope Return Path)
 - Mailing lists, 35–14, 39–138
 - viaaliasoptional channel option, 33–44
 - viaaliasrequired channel option, 33–44, 52–1
 - Error text if user not found, 39–158
 - ims-ms channels, 51–3
 - viametermaid pwexpirealert option, 21–13
 - Virtual domain (vdomain) options, 27–3
 - authcachettl, 27–5
 - authenticationldapattributes, 27–6
 - authservice, 27–6
 - authservicettl, 27–6
 - clientlookup, 27–9
 - crams, 27–11
 - debugkeys, 27–11
 - defaultdomain, 27–12
 - domainsearchformat, 27–14
 - ehlokeywords, 27–14
 - failovertimeout, 27–15
 - hosteddomains, 27–15
 - ldapcachesize, 27–16
 - ldapcachettl, 27–16
 - mailhostattrs, 27–18
 - popbeforesmtpkludgechannel, 27–20
 - preauth, 27–20
 - replayformat, 27–20
 - restrictplainpasswords, 27–21
 - searchformat, 27–21
 - smtpproxypassword, 27–22, 39–218
 - ssladjustciphersuites, 7–11, 27–24
 - storeadmin, 27–27
 - storeadminpass, 27–27
 - tcpaccess, 27–28
 - tcpaccessattr, 27–28
 - virtualselecteddelim, 27–29
 - virtualselecteddelim MMP/IMAP Proxy/POP Proxy/vdomain option, 27–29
 - virtualselectedfile MMP/IMAP Proxy/POP Proxy/SUBMIT Proxy option
 - DELETED; see vdomain options instead, 27–29
 - VMS MAIL user agent
 - header* channel options, 33–69
 - vms_mail_exclusive MTA option, 39–273
 - vrifyallow channel option, 33–134
 - vrifydefault channel option, 33–134
 - vrifyhide channel option, 33–134
- ## W
- warninginterval alarm.system:diskavail option, 11–3
 - warninginterval alarm.system:serverresponse option, 11–3
 - warningthreshold msprobe option, 10–2
 - warnpost channel option, 33–94, 47–1
 - Watcher, 1
 - Options, 9–1
 - enable, 9–1
 - port, 9–1
 - secret, 9–1
 - Startup, 9–1
 - stressperiod base option, 7–5
 - welcomemsg base option, 7–17
 - welcomemsg message_language option, 17–1
 - withinresolution IMAP option, 21–5
 - workday Message Store deleted expire option, 16–27
 - wrapsmtplonglines channel option, 33–133
- ## X
- xclient channel option, 33–76, 33–132, 33–153

xclientrepeat channel option, 33–76, 33–132, 33–153
xclientsasl channel option, 33–76, 33–132, 33–153
xclientsaslrepeat channel option, 33–76, 33–132,
33–153
xmailer MSHTTP option, 29–6
x_env_to channel option, 33–75

Colophon

This manual was automatically generated from Messaging Server source code. A series of custom stylesheets were used to produce DocBook 5 source, which was then processed using the DocBook XSL stylesheets to produce XSL:FO source. Finally, the XSL:FO source was processed using Apache FOP to produce a Portable Document Format file.

The titles in this document are typeset in Helvetica; the body text is in Palatino.

