Oracle Communications Messaging Server Reference

Release 8.1

F15150-02

July 2020

ORACLE

Oracle Communications Messaging Server Reference

Release 8.1
F15150-02

Contents

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 Installation and Configuration Guide for Cassandra Message Store*
- *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.*, unix user ids) are stored instead in the `restricted.cnf` file. However, under normal circumstances, the Messaging Server unified configuration file `config.xml` is not---indeed should not be--- inspected or editted 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 langauge.

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

Option settings in Unified Configuation, 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 post
  - Host list
  - IPv4 range

- *File and directory types*

  - MTA-specific directory path
  - MTA-specific file path
  - MTA-specific file 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:

`P`*`year`*`Y`*`month`*`M`*`week`*`W`*`day`*`DT`*`hour`*`H`*`minute`*`M`*`second`*`S`

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.

Note that all of the letters in ISO 8601 P values must be written in upper case.

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

Besides the `backoff` channel option and a few alias options, also several MeterMaid `local_table` 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-dd``T`*`hhmmss.ss`*`Z`

or with an optional time zone offset:

`yyyy-mm-dd``T`*`hhmmss.ss`*`+`*`hh:mm`*

or

`yyyy-mm-dd``T`*`hhmmss.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.

The `imsimta test -time` utility may be used to test the validity of an ISO 8601 time string, as well as convert it into the (perhaps more familiar) date-time format used in RFC 822 header fields, such as Received: and Date: header fields.

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:`
- `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

- `redis:` -- new in MS 8.0.2.3

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" (`pabldap:`) 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 external directory lookup MTA options, LDAP PAB MTA options, BURL MTA options, Memcache MTA options, MeterMaid MTA options, or Redis 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 `hashdir` algorithm to *`string`* 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*`table-name`*, probing with *`argument`*; if the mapping table is found and 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
`$`*`n`*`A`	Insert the *`n`*th character of the Address
`$B`	LDAP user root; *i.e.*, the value of the `ugldapbasedn` base option (in legacy configuration, the `local.ugldapbasedn` configutil parameter), or as

	overridden by the MTA-specific `ldap_user_root` MTA option
`$C`	LDAP domain root; *i.e.*, the value of the `dcroot` base option (in legacy configuration, the `service.dcroot` configutil parameter), or as overridden by the MTA-specific `ldap_domain_root` MTA option
`$D`	Domain name
`$E`	Synonymous with `$1E`; that is, substitute in the value of the LDAP attribute named by `ldap_spare_1`
`$`*n*`E`	Substitute in the value of the LDAP attribute named by the `ldap_spare_n` MTA option, where *n* 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.
`$F`	Delivery filename; some syntax checking is done if the "name" is a file URL (`file:` syntax)
`$G`	Synonymous with `$2G`; that is, substitute in the value of the LDAP attribute named by `ldap_spare_2`
`$`*n*`G`	Similar to `$`*n*`E`, but with the default for *n* being 2 rather than 1
`$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 `service.defaultdomain` configutil parameter), or the MTA-specific override option `ldap_default_domain`; when the host domain matches the `defaultdomain`, 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 `service.defaultdomain` configutil parameter), or the MTA-specific override option `ldap_default_domain`; when the host domain matches the `defaultdomain`, 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 `service.defaultdomain` configutil parameter), or the MTA-specific override option `ldap_default_domain`; when the host domain matches the `defaultdomain`, then no domain is inserted and also the leading character is removed. This substitution is typically used to remove a leading

	percent character, %, 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 service.defaultdomain configutil parameter), or the MTA-specific override option ldap_default_domain; when the host domain matches the defaultdomain, 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, /, 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 ldap_user_object_classes and ldap_group_object_classes MTA options
$L	Username minus any special leading characters such as ~ or _
$M	uid; when the LDAP URL relates to a user address, this would be the actual uid value (or whatever LDAP attribute is named by the ldap_permid MTA option, and if that attribute is not present, the attribute named by the ldap_uid MTA option), but in other contexts may have another meaning (as for instance, $M substitutes the detailed verdict string for spam/virus filter package integration spamfilter*N*_action_*M* MTA options)
$*n*M	New in MS 6.2. Insert the *n*th character of the uid (or an " a" if the uid has no *n*th character)
$N	Construct a comma-separated list of attributes (normally those to return in reverse_url lookups, hence a comma separated list of the attributes named by the ldap_capture, ldap_recipientlimit, ldap_recipientcutoff, ldap_sourceblocklimit, ldap_preferred_language, ldap_personal_name, ldap_primary_address, and ldap_equivalence_addresses MTA options, as well as (in MS 6.2 and later) ldap_source_conversion_tag, and (in MS 6.3-0.15 and later), ldap_blocklimit, ldap_source_channel, the ldap_source_optin*N* MTA options, ldap_preferred_country, and the ldap_spare_*N* MTA options)
$O	(New in MS 6.1p1/6.2) Substitute the source route
$4O	(New in 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---nonexistent---source route)
$P	Program name, that is, the value of the attribute named by the ldap_program_info MTA option (hence normally the value of the mailProgramDeliveryInfo attribute)
$Q	Filter for address reversal in iMS 5.2p2; that is, a filter constructed from the attribute names listed in the ldap_mail_reverses MTA option (or the appropriate default attribute names, depending upon the schema tag, if ldap_mail_reverses is not set); this substitution sequence was typically used as the filter in reverse_url lookups in iMS 5.2, but with the Messaging Server 6.0 MTA, the $R substitution sequence is normally used instead
$S	Subaddress
$2S	Subaddress if present, but if there is no subaddress, then remove the leading character (used to remove the plus character, +)
$R	Filter for mail aliases; that is, a filter constructed from the attribute names listed in the ldap_mail_aliases MTA option (or with legacy configuration in the local.imta.mailaliases configutil parameter); as of MS 6.0 and later, also typically used as the search filter for reverse_url address reversal lookups
$T	Turn domain name *chunk1.chunk2...chunkn* into dc=*chunk1*,dc=*chunk2*,...,dc=*chunkn*
$U	The username portion of the address (that is, the local-part sans subaddress and sans any leading backwards-single-quote, tilde, or underscore character)
$*n*U	Insert the *n*th character of the username
$V	The base DN returned by calling domainMap on the domain in the address; note that this can be affected by the domain_uplevel MTA option
$1V	A variant on the $V substitution, where if the $V domainMap sort of lookup fails (no inetDomainBaseDn or aliasedObjectName matching the domain name is found), then this $1V substitution instead returns the DN of the top of the user and group tree, that is, the value of the ugldapbasedn base option (in legacy configuration, the local.ugldapbasedn configutil parameter) or as overridden via the MTA-specific ldap_user_root MTA option
$X	mailHost
$*n*X	Insert the *n*th component of the mailHost

Chapter 2 Options for migrating to Unified Configuration

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. It may also be used briefly when adding a new system to an existing deployment via the Deployment Map server. In that setting, the Deployment Map client briefly runs, registering the node with the server and reporting as its role, the string value set with this `rolename` option.

2.3 `plugins` Option

The `plugins` option enables 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. (These names tend to correspond to former, now obsolete, MTA Tailor options; as such, while the MTA Tailor options per se no longer exist for purposes of *setting* directory locations, file names, or values, these corresponding special symbolic names can also still generally be used in MTA contexts, where they now are interpreted as having values relative to SERVERROOT.)

```
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 SERVERROOT value is used to construct the DATAROOT value (either `/var/`SERVERROOT or SERVERROOT`/data`) and CONFIGROOT value (either `/var/`SERVERROOT`/config` or SERVERROOT`/config`).

Chapter 4 Recipe language

Recipe files are used to automate configuration management tasks within the `msconfig` utility. Recipe files are expressed using a domain specific 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 inspirations for the recipe language is the Icon programming language designed by Ralph Griswald as well as the IETF's Sieve mail filtering language.

Recipe file syntax includes C-like expressions, operators, and assignments, Sieve-like conditionals, and loops. The available data types are integers, strings, and lists.

Like `msconfig` itself recipes can only be used with a unified configuration.

Note that `imsimta test -expression -xc` can be used to test the syntax of recipe operations.

Recipes are typically written to operate in several phases:

1. Checks are done to make sure the right conditions exist for the recipe to be effective. If the desired conditions aren't met the recipe can either issue a warning or exit with an error.

2. The recipe asks a number of questions to determine exactly what changes should be made.

3. The recipe optionally uses the `continue` function to make sure the user wishes to proceed if any warnings were issued.

4. Finally, the recipe implements 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 String values 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
"\uNNNN" - Unicode character, must specify exactly 4 hex digits
           (new in MS 8.0.2.3)
```

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

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

As of MS 8.1.0.1 trailing commas are allowed in lists, making them easier to maintain:

```
["c1", "c2", ]
```

Note that the elements of a list are always strings.

4.3.1 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 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 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,i)
```

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"]
```

Note that parentheses may be used in place of square brackets for indexing.

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);
      ...
      nextif (expression);
      ...
     }
```

A loop may contain zero or more `exitif` and/or `nextif` statements. The loop terminates if the argument to `exitif` evaluates to `true`. New in 8.0.2.3, `nextif` can be used to cause the loop to restart if the associated condition evaluates to `true`.

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 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
?*v*	Force interpretation of *v* as a variable, rather than as a pre-defined function	19
***f*(...)**	Function call	18
Index		
***sl*[*i*]**	Return the *i*th character of string/list *sl*+	17
***sl*[*i*,*j*]**	Return the *i*th through *j*th characters of string/list *sl*+	17
***sl*(*i*)**	Return the *i*th character of string/list *sl*	17
***sl*(*i*,*j*)**	Return the *i*th through *j*th characters of string/list *sl*	17
Increment/decrement		
***v*++**	Return *v* and then increment its value (*v* must be a variable with an integer value)	16
***v*--**	Return *v* and then decrement its value (*v* must be a variable with an integer value)	16
++*v*	Increment the value of *v* and then return it (*v* must be a variable with an integer value)	16
--*v*	Decrement the value of *v* and then return it (*v* must be a variable with an integer value)	16
Unary bitwise and logical		
~*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 **<<** *m*	Left shift	11
n **>>** *m*	Right shift	11
Comparisons		
n **<** *m*	Less than	10
n **<=** *m*	Less than or equal to	10
n **>=** *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 **^** *m*	Bitwise xor	7
n **|** *m*	Bitwise or	6
n **contains** *m*	String/list *n* contains string *m*. The position of the first match in the string is returned if *n* is a string; the position of the first matching string in the list is returned if *n* is a list.	5
n **matches** *m*	String/list *n* is matched by the glob pattern*m* true/false is returned if *n* is a string, the position of the first matching string in the list is returned if *n* is a list.	5
Infix logical		
n **&&** *m*	Logical and	4
n **^^** *m*	Logical xor	3
n **||** *m*	Logical or	2
Conditional		
p **?** *n* **:** *z*	*n* if *p* is nonzero, *z* otherwise	1
p **? :** *z*	(New in MS 8.1.0.3) *p* if *p* is nonzero, *z* otherwise	1
p **?** *n*	(New in MS 8.1.0.3) *n* if *p* is nonzero, *p* (0) otherwise	1
Assignment		
v **=** *e*	Assign *v* the value of *e* (*v* must be a variable)	0
v **+=** *e*	Add/concatenate the value of *e* to *v* (*v* must be a variable)	0
v **-=** *e*	Subtract the value of *e* from *v* (*v* must be a variable with an integer value)	0
v ***=** *e*	Multiply/cross product *v* by the value of *e* (*v* must be a variable)	0
v **/=** *e*	Divide *v* by the value of *e* (*v* must be a variable with an integer value)	0

v **&=** *e*	Bitwise and *e* into *v* (*v* must be a variable with an integer value)	0
v **|=** *e*	Bitwise or *e* into *v* (*v* must be a variable with an integer value)	0
v **^=** *e*	Bitwise xor *e* into *v* (*v* must be a variable with an integer value)	0
v **<<=** *e*	Left shift *v* by *e* (*v* must be a variable with an integer value)	0
v **>>=** *e*	Right shift *v* by *e* (*v* must be a variable with an integer value)	0
v **.=** *e*	Concatenate the value of *e* onto *v* (*v* must be a variable)	0
v1 **:=:** *v2*	Exchange the values in *v1* and *v2* (*v1* and *v2* must both be variables)	0

[+] Most operators are available for use in Sieve filters, as well as in recipes, *except* for use of square brackets, `[...]`, for indexing into strings or lists. Sieve filter syntax uses square brackets to indicate Sieve lists, so in Sieve filters the alternate syntax for substring indexing of `(...)` must be used.

4.7 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 recipe-specific functions in groups. For general string/list/integer functions, see Symbol table functions.

Table 4.2 Alphabetical List of Built-in Functions

Function	Description
abs(*i***)**	Return the absolute value of the numeric value *i*.
add_alias(*s*,*c*[,*e*]**)**	New in 8.0.1.2. Adds an alias named *s* containing the optlist *c* and, optionally, additional alias entries from the list *e*. An error will be returned if the alias already exists. Each element of the optlist specifies an alias option and its corresponding value. An empty string must be specified for alias options that do not accept a value.
add_channel(*s*,*c***)**	Adds a channel named *s* containing the optlist *c*. 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.
add_group(*g*,*c***)**	Adds a group named *g* containing the optlist *c*. An error will be returned if the group already exists. Each element of the optlist specifies a group element and its corresponding value.
add_mapping(*s*,*c***)**	Adds a mapping named *s* containing the optlist *c*. 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.
allof(*i1*[,*i2*...]**)**	Returns a nonzero (true) value if all of *i1*, *i2*, ... are nonzero; returns `0` otherwise.

any(*s1*,*s2*)**	Return 2 if any character in *s1* appears as the first character of *s2*; return 0 otherwise.
anyof(*i1*[,*i2*...]**)**	Returns a nonzero (true) value if any of *i1*, *i2*, ... are nonzero, returns 0 if all are zero.
append_alias(*s*,*c*[,*e*]**)**	New in 8.0.1.2. Append the alias options specified in the optlist optlist *c* and, optionally, the alias entries specified by the list *e* to the alias named by *s*. The alias will be created if it doesn't already exist.
append_group(*s*,*c***)**	Appends the contents of optlist *c* to the group named *s*. The group will be created if it doesn't already exist. Each element of the optlist specifies a group element and its corresponding value.
append_mapping(*s*,*c***)**	Appends the contents of optlist *c* to the mapping named *s*. 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.
append_rewrites(*c***)**	Appends the contents of optlist *c* 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.
argc	(New in 8.0) Returns the number of additional arguments given to the `msconfig run` command.
argv(*n***)**	(New in 8.0) Returns the *n*th argument given to the the `msconfig run` command as a string. The value *n* must be between 1 and **argc** inclusive.
bal(*c1*,*c2*,*c3*,*s***)**	Scan *s* looking for an occurrence of a character in *c1* that is balanced with respect to *c2* and *c3*. Returns the position of the first balanced *c3* character in *s* if one is found, `length(s)+1` if no *c3* character is found but the string as a whole is balanced, or 0 if the string isn't balanced.
call_user(*s1*,*s2*,*e*[,*e2*...]**)**	Call the routine named *s2* in the external library image *s1*, passing the argument *e* (and optionally additional, comma-separated arguments).
chr(*i1*[,*i2*...]**)**	Returns a string containing successive characters with decimal values *i1*, *i2*, ...
continue([*s1*[,*s2*]]**)**	The `continue` 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 *s1*. 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 *s1* is specified. An optional argument *s2* may be supplied as a default response to `continue`; for instance, `continue("Continue [y]? ","Y")` will mean that a user empty (carriage return) response will mean to continue.
continue	The `continue` 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, and `continue` was not provided with a prompt string, then the administrator is prompted with the default prompt "`Some warnings have occurred. Do you want to continue`

	`[N]?` ". 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.
decode *[:e] s*	Decodes the string *s* from the specified encoding *:e*. The *:e* nonpositional parameter must be one of :base64, :base85, :hex, :idn, or :quotedprintable. The default is :hex.
default	Undoes the effect of `instance` or `role`: options are interpreted as being of their preferred flavor.
defined(*s***)**	Returns `1` if *s* is defined as a variable; return `0` otherwise.
delete_alias(*s***)**	New in 8.0.1.2. Delete the alias named *s*. No operation is performed if the alias does not exist.
delete_channel(*s***)**	Delete the channel named *s*. No operation is performed if the channel does not exist.
delete_file(*s***)**	New in 8.0.1.2. Delete the file named *s*. Returns `1` if the delete operation succeeded, and `1` if not.
delete_group(*s***)**	Delete the group named *s*. No operation is performed if the group does not exist.
delete_mapping(*s***)**	Delete the mapping named *s*. No operation is performed if the mapping does not exist.
delete_options(*l***)**	Deletes all values associated with the options specified in the list *l*.
delete_optlist(*o*,*n*...**)**	Delete option *n* from optlist *o* 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. As of MS 8.0.1.3, *n* can be a two element list specifying a name/value pair to be removed from the list.
delete_rewrites(*l***)**	Delete the rewrite rules specified in list *l*. No operation is performed if such rewrite rules do not exist.
delete_statefile(*v***)**	Delete the specified statefile variable *v*. The function returns `true` if the delete operation was successful; `false` if statefile support is not enabled.
deploymap ...	(New in MS 8.0.1.1.0) Perform various operations on and obtain data from the deployment map.
deploymap :add *[***:deployment** *d]* **:host** *h [***:role** *r]*	(New in MS 8.0.1.1.0) Add host(s) *h* with role *r* to deployment *d* in the deployment map. *h* may be either a string or list. An error occurs if any of the hosts already exist. The current deployment is used if *d* is not specified; if there is no current deployment, then the first deployment in the deployment map is selected; a deployment with the default name "`site-01`" will be created if no deployment exists in the deployment map. No role with be associated with the host(s) if *r* is not specified. Returns the number of modifications that were made to the deployment map.
deploymap :add *[***:deployment** *d]* **:host** *h* **:property** *p*	(New in MS 8.0.1.1.0) Add property/properties *p* to host *h* in deployment *d*. Host *h* will be created if it does not already exist. The current deployment is used if *d* is not specified; if there is no current deployment, then the first deployment in the deployment map is selected; a deployment with the default name "`site-01`" will be

	created if no deployment exists in the deployment map. Returns the number of modifications that were made to the deployment map.
deploymap :add *[***:deployment** *d]* **:host** *h* **:property** *p* **:role** *r*	(New in MS 8.0.1.1.0) The host *h* is created in deployment *d* with role *r* and properties *p*. The host must not already exists. The current deployment is used if *d* is not specified; if there is no current deployment, then the first deployment in the deployment map is selected; a deployment with the default name "`site-01`" will be created if no deployment exists in the deployment map. Returns the number of modifications that were made to the deployment map.
deploymap :create	(New in MS 8.0.1.1.0) Create a new, empty deployment map. Note that if you write this out using `msconfig`'s `DEPLOYMAP WRITE` command you will delete all your existing entries!
deploymap :delete :deployment *d*	(New in MS 8.0.1.1.0) Delete the deployment named by string *d* from the deployment map. Deleting a nonexistent deployment is a no-op. Returns the number of modifications that were made to the deployment.
deploymap :delete *[***:deployment** *d]* **:host** *h*	(New in MS 8.0.1.1.0) Delete host(s) *h* from deployment *d*. *h* can be either a string or list. Deleting a nonexistent host is a no-op. The current deployment is used if *d* is not specified; if there is no current deployment, the first deployment in the deployment map is selected. Returns the number of modifications that were made to the deployment map.
deploymap :delete *[***:deployment** *d]* **:host** *h* **:role** *r*	(New in MS 8.0.1.1.0) Delete any role associated with host(s) *h* in deployment *d*. Deleting a nonexistent role is a no-op. An error occurs if host *h* does not exist. The current deployment is used if *d* is not specified; if there is no current deployment, the first deployment in the deployment map is selected. Returns the number of modifications that were made to the deployment map.
deploymap :delete *[***:deployment** *d]* **:host** *h* **:property** *p*	(New in MS 8.0.1.1.0) Delete any properties for host(s) *h* (which may be a string or list) in deployment *d* that match the glob-wildcarded values in string or list*p*. The current deployment is used if *d* is not specified; if there is no current deployment, the first deployment in the deployment map is selected. Returns the number of modifications that were made to the deployment map.
deploymap :dump	(New in MS 8.0.1.1.0) Returns a string containing an outline of the entire deployment map.
deploymap :list...	(New in MS 8.0.1.1.0)
deploymap :read *s*	(New in MS 8.0.1.1.0) Reads a JSON-format deployment map from string *s*.
deploymap :rename *s* **:deployment** *d*	(New in MS 8.0.1.1.0) Renames deployment *d* to *s*. *d* must specify an existing deployment; *s* must be a nonempty utf-8 string. Returns the mumber of modifications that were made to the deployment map.
deploymap :rename *[***:deployment** *d]* *s* **:host** *h*	(New in MS 8.0.1.1.0) Renames host *h* in deployment *d* to *s*. The current deployment is used if *d* is not specified; if there is no current deployment, then the first deployment in the deployment map is selected. Host *h* must already exist and *s* must be a valid domain name. Returns the mumber of modifications that were made to the deployment map.

deploymap :set *[*:**deployment** *d]* **:host** *h* **:role** *r*	(New in MS 8.0.1.1.0) Sets the role for host(s) *h* in deployment *d* to *r*. *h* can be either a string or list. The specified host(s) *h* must already exist in the deployment. The current deployment is used if *d* is not specified; if there is no current deployment, then the first deployment in the deployment map is selected. Returns the number of modifications that were made to the deployment map.
deploymap :write	(New in MS 8.0.1.1.0) Returns the contents of the current deployment map as a JSON-formatted string. Note that no mechanism is provided in the recipe language to update the active deployment map; this can only be done at the `msconfig` level.
description(*d***)**	The `description` function is used to provide a description of the function of a recipe. The value of the string argument *d* is displayed when the recipe is processed by the `msconfig directory` command. Note that *d* must be a literal string enclosed in quotes; it cannot be an expression. When the recipe is executed the `description` function does nothing other than return the value of its argument *d*.
edit(*s***)**	Place the value of the string *s* in a temporary file and invoke the external editor specified by the `EDITOR` environment variable on the file. When the editor returns the `edit` function returns the (possibly modified) file content.
encode *[:e] s*	Encodes the string *s* into the specified encoding *:e*. The *:e* nonpositional parameter must be one of :base64, :base85, :hex, :idn, :param, :quotedprintable, or :url. The default is :hex.
error(*s***)**	Issue error string *s* back to administrator executing the recipe.
exists_alias(*s***)**	New in 8.0.1.2. Returns a nonzero integer (specifically, the number of entries in the alias group) if the alias named *s* exists, zero if it does not.
exists_channel(*s***)**	Returns a nonzero integer (specifically, the number of channel options set on the channel) if the channel named *s* exists, zero if it does not.
exists_file(*s***)**	Returns `1` if the file named by *s* exists, `0` if it doesn't. If only a file name is specified with no path, `msconfig` looks in the MTA configuration directory, `IMTA_TABLE:`.
exists_group(*s***)**	Returns a nonzero integer if the group named *s* exists, zero if it does not.
exists_mapping(*s***)**	Returns a nonzero integer if the mapping named *s* exists, zero if it does not.
exists_option(*s***)**	Returns the number of values currently set for the option *s*. A value of `0` is returned if the option isn't set.
exists_optlist(*o***,***s***)**	Returns `1` value if the the optlist *o* contains an option named *s*, `0` otherwise.
exists_statefile(*v***)**	Returns 1 if the statefile variable named *v* is defined, 0 if it is not, and -1 if statefile support has not been enabled with the `--statefile` switch.
find(*s1***,***s2***[,***i***,***j***])**	Returns the position of the first occurrence of *s1* in *s2*`[`*i*`:`*j*`]`. The entire string is searched if `i` and `j` are omitted.
find(*s***,***l***[,***i***,***j***])**	Returns the position of the first list element from *l*`[`*i*`:`*j*`]` that matches *s*. The entire list is searched if `i` and `j` are omitted.

get_alias(*s*)	New in 8.0.1.2. Returns the definition (alias options) of the alias named *s* as an optlist. An empty list is returned if the alias does not exist.
get_channel(*s*)	Returns the definition (channel options) of the channel named *s* as an optlist. An empty list is returned if the channel does not exist.
get_default(*s*)	Returns the built-in default for the specified option *s* as a string. An empty string is returned if the option has no default. A error occurs if the option does not exist. New in MS 8.0.2.1.
get_group(*s*)	Returns the content of the group named *s* as an optlist. An empty list is returned if the group does not exist.
get_mapping(*s1*[,*s2*[,*s3*]])	Returns the content of the mapping named *s1* as an optlist. An empty list is returned if the mapping does not exist. Optionally, strings *s2* (string with wildcards to match patterns) and *s3* (string with wildcards to match templates) may be specified, to restrict which entries of the named mapping to return. If not specified, *s2* and *s3* each default to "*", meaning that *all* of the entries of the named mapping will be returned.
get_msconfig_info(*s*)	New in MS 8.0.2.3. Return the value of the msconfig information item *s*. An error occurs if an unknown item is specified. Note that *s* is not case sensitive.
get_option(*s*)	Returns the value of the option named *s* as a string. An error will occur if the specified option name is invalid or matches multiple options. An empty string is returned in two cases: if the specified option is a valid no-value option that is set, or if the specified option is valid but not set.
get_option_modification_locations	(New in 8.0.6) Returns a list of locations in the current recipe where a configuration modification occurred.
get_options(*s*)	Returns the names and values of the options named *s* as an optlist. (Note that when an option is set in both role and instance flavors, only the instance is returned.) An empty list is returned if the specified option is valid but not set.
get_optlist(*o*,*s*)	Returns the value of the option named *s* from the optlist *o* as a string. An empty string is returned if the specified option is not in the optlist.
get_path(*r*)	(New in 8.0) Returns a path to a directory in the server instance. The argument *r* can be one of "`server`", "`data`", or "`config`", which will return the path to the server root, the data root, or configuration root directories, respectively.
get_rewrites[(*p*[,*t*])]	Returns the selected set of rewrite rules as an optlist. *p* 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, *t* 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.
get_statefile(*v*)	Returns the value of the statefile variable named *v*. An empty string is returned if the specified variable is not in the statefile or statefile support has not been enabled with the `--statefile` switch.
get_system_info(*s*)	New in 8.0.1.2. Return the value of the system information item *s*. An error occurs if an unknown item is specified. Note that *s* is not case sensitive.

getenv(*s***)**	Return the value of the environment variable *s*. An empty string is returned if *s* is not defined. Note that *s* is case sensitive.
hash *[:e] [:h] v*	Returns the hash of the value *v*. *v* 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 *:h* may be any of `:md2`, `:md4`, `:md5`, `:sha1`, `:sha256`, `:sha512`, `:ripemd128`, or `:ripemd160`; the default is `:sha1`. (Support for `:sha256` and `:sha512` is new in MS 8.0.2.3.) An encoding may optionally be applied; *:e* may be any of `:hex`, `:quotedprintable`, `:base64`, `:base85`, or `:binary`. `:binary`, or no encoding, is the default.
hash_hmac *[:e] [:h] k v*	Returns the hmac of the value *v* using key *k*. *v* 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 *:h* may be any of `:md2`, `:md4`, `:md5`, `:sha1`, `:sha256`, `:sha512`, `:ripemd128`, or `:ripemd160`; the default is `:sha1`. (Support for `:sha256` and `:sha512` is new in MS 8.0.2.3.) An encoding may optionally be applied; *:e* may be any of `:hex`, `:quotedprintable`, `:base64`, `:base85`, or `:binary`. `:binary`, or no encoding, is the default.
instance	Operate on `instance` options.
integer(*e***)**	Converts *e* to an integer. If *e* is already an integer, it is returned unchanged; if *e* is a string, it is read as a sequence of ASCII digits. If *e* is a list, it must contain one element and that element is treated in the same way a string would be.
keywords(*l***)**	The `keywords` function is used to provide a list of keywords associated with a recipe. The value of the list argument *l* is displayed when the recipe is processed by the `msconfig directory` command.
lcase(*e***)**	Converts any upper case characters in *e* to lower case. If *e* is a number, it is converted to a string.
ldap_init(*o***)**	(New in MS 8.0.1) Initializes the built in LDAP client. This call must be performed prior to using any of the other `ldap_*` functions. The client uses the current settings of the configuration options `ugldaphost`, `ugldapbinddn`, `ugldapbindcred`, `ugldapport`, and `ugldapusessl` when it initializes. The single argument *o* 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.
ldap_ldif(*s***[,***f***])**	(New in MS 8.0.1) Apply the LDIF specified in the string *s* 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 *f* 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 4096) - if set, treat any LDAP error that occurs as a recipe error. An integer count of the number of successful modifications performed is returned. `ldap_init` must be called before calling `ldap_ldif`.
`ldap_search(`*`o[,n]`***`)`**	(New in MS 8.0.1.1.0) The `ldap_search` function takes an optlist argument specifying at least the `basedn` for the search, and optionally also the attribute to return (`attrs`), the `scope` of the search, and a filter (`filter`) for the search. Valid values for the "`scope`" are: `base`, `onelevel` (or `one`), `subtree` (or `sub`). An optional second integer argument specifies the number of entries to return, and defaults to `-1` (return all entries) if omitted. The function returns an optlist containing the attribute-value pairs matching the search.
`left(`*`s1,i[,s2]`***`)`**	Returns the leftmost *`i`* characters of *`s1`*. If *`i`* is greater than `length(`*`s1`*`)` the result is padded with *`s2`*. As much of *`s2`* as is necessary will be used; if *`s2`* is too short it will be used multiple times. *`s2`* defaults to a space if it is omitted.
`left(`*`l1,i[,l2]`***`)`**	Returns the leftmost *`i`* elements of *`l1`*. If *`i`* is greater than `length(`*`l1`*`)` the result is padded with *`l2`*. As much of *`l2`* as is necessary will be used; if *`l2`* is too short it will be used multiple times. *`l2`* defaults to one empty list element if it is omitted.
`length(`*`s`***`)`**	Returns the number of 8-bit characters in the string *`s`*.
`length(`*`l`***`)`**	Returns the number of elements in the list *`l`*.
`list(`*`s,n`***`)`**	Returns a list *`n`* elements long with each element equal to *`s`*.
`list(`*`l,n`***`)`**	Returns a list consisting of *`n`* copies of *`l`*.
`list_names(`*`s[,n]`***`)`**	Returns the options whose name begins with the specified string *`s`* as an optlist. The optional integer second argument specifies whether or not to retain backslash characters in the options' values; the default is `0` (false).
`make_path(`*`p`***`)`**	Converts a path *`p`* using `IMTA_TABLE:` 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 `msconfig` is running.
`match(`*`r,s`***`)`**	Returns `1` (true) if the regular expression *`r`* matches a substring of string *`s`*, `0` (false) otherwise. Note that the pattern *`r`* 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.
`map(`*`s1,s2,s3`***`)`**	Returns a string obtained by mapping characters of *`s1`* that occur in *`s2`* into corresponding characters in *`s3`*. Characters that don't appear in *`s2`* are unchanged.
`max(`*`i,j[,...]`***`)`**	Returns the largest element in a set of integers.
`max(`*`s1,s2[,...]`***`)`**	Returns the largest element in a set of strings.
`min(`*`i,j[,...]`***`)`**	Returns the smallest element in a set of integers.
`min(`*`s1,s2[,...]`***`)`**	Returns the smallest element in a set of strings.
`ord(`*`s`***`)`**	Returns the ordinal value of the first character of *`s`*, which must contain at least one character.

pop(*l***)**	Returns the first element of list *l*. The element is deleted from the list. *l* must be a variable with a list value.
prepend_alias(*s***,***c***[,***e***])**	New in 8.0.1.2. Prepends the contents of optlist *c*, and, optionally, the alias entries from the list *e*, to the alias named *s*. The alias will be created if it doesn't already exist.
prepend_group(*s***,***c***)**	Prepends the contents of optlist *c* to the group named *s*. The group will be created if it doesn't already exist.
prepend_mapping(*s***,***c***)**	Prepends the contents of optlist *c* to the mapping named *s*. The mapping will be created if it doesn't already exist.
prepend_rewrites(*c***)**	Prepends the contents of optlist *c* to the rewrite rules.
print(*s***)**	Print the string *s* on the administrator's terminal.
push(*l***,***s***)**	Adds the string *s* to the beginning of list *l*. The list *l* is updated as well as being returned. *l* must be a variable with a list value.
put_optlist(*o***,***s***,***v***...)**	Put the option *s* with the value *v* in optlist *o* 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 put multiple options on the optlist.
random(*n***)**	Returns a random integer value between 0 and *n-1*.
randomseed(*n***)**	Seeds the random number generater function random with the integer seed value *n*.
read(*s1***[,***s2***[,***s3***]])**	Read and return a string from the administrator terminal, prompting with the string *s1*. The optional string argument *s2* provides a default value for the function to return, if the administrator does not enter an explicit value (merely enters a null response). New in MS 8.0.2.3, the string argument *s3* specifies the name of a statefile variable whose value is used if present or will be updated with whatever value is entered.
read_file(*s***)**	Reads and returns the content of the file named by *s*. `msconfig` looks in the MTA configuration directory, `IMTA_TABLE:`, if *s* specifies only a file name with no path. An error occurs if the file doesn't exist or cannot be read. Line feeds are used as line separators.
read_optlist(*s***)**	Converts a string *s* containing options into an optlist which is returned. The string *s* should be formatted with options expressed as *name=value* delimited by CR or LF. Leading space or horizontal tab characters prior to a *name* will be ignored (though space and horizontal tab are allowed and retained within a *value*). Comments, which start with exclamation point `!` or hash character #, will be ignored.
read_password *[***:minlength** *n1] [***:maxlength** *n2] [***:upper** *n3] [***:lower** *n4] [***:digit** *n5] [***:symbol** *n6] [s1] [s2]*	Read a password from the administrator terminal, returning it as a string. Optional parameters may be used to specify requirements that the password must meet: `:minlength` default is `-1` (no minimum), `:maxlength` default is `-1` (no maximum), by default no requirements are placed on number of upper case characters, lower case characters, digits, or symbols (respectively, `:upper`, `:lower`, `:digit`, `:symbol`). The administrator is prompted with string *s1* (default "`Password: `") and optionally may be prompted to verify the password with string *s2* (default "`Verify `").
repl(*s***,***j***)**	Returns a string consisting of *j* concatenatations of *s*.

repl(*l*,*j***)**	Returns a list consisting of *j* concatenatations of *l*.
replace_alias(*s*,*c*[,*e*]**)**	New in 8.0.1.2. Replaces the alias named *s* with the contents of optlist *c* and, optionally, alias entries from the list *e*. The alias will be created if it doesn't already exist.
replace_channel(*s*,*c***)**	Replaces the channel named *s* with the contents of optlist *c*. The channel will be created if it doesn't already exist.
replace_group(*s*,*c***)**	For the group named by string *s*, replace its options with those in the optlist *c*.
replace_mapping(*s*,*c***)**	Replaces the contents of the mapping named *s* with the contents of optlist *c*. The mapping will be created if it doesn't already exist.
replace_rewrites(*c***)**	Replaces the current rewrite rules with the rewrite rules specified in the optlist *c*.
resolve_option(*o***)**	The string value *o* is run through the option name resolution process and the resolution results are returned as a bit-encoded integer. The option need not exist in the current configuration and is not modified in any way. Currently four bits are defined: Bit 0 (value 1), if set, indicates the path specified was valid, bit 1 (value 2), if set, indicates the name specifies a valid option (as oppose to, say, the name of an option group), bit 2 (value 4), if set, indicates that the option has been deleted and is effectively a no-op, bit 3 (value 8), if set, indicates that the option is restricted, and bit bit 4 (value 16), indicates that this is an instance-only option.
restricted(*p***)**	The `restricted` function is used to control or check the recipe's ability to modify the values of restricted options. If the `-restricted` switch was given to the `run` command, the `restricted` function returns a value of `1` (true) unconditionally. If `-restricted` wasn't specified, the argument *p* will be evaluated to determine if the user should be prompted. A zero value will return a value of `0` (false). A nonzero value will prompt the user for permission to modify restricted options. A response beginning with "`Y`", "`y`", "`T`", "`t`", or "`1`" will enable modification access to restricted options and the function will return a value of `1` (true). A response beginning with "`N`", "`n`", "`F`", "`f`", or "`0`" will return a value of `0` (false). Any other response will cause the prompt to be repeated.
reverse(*s***)**	Reverses all the characters in string *s* and returns the result.
reverse(*l***)**	Reverses all the elements in list *l* and returns the result.
right(*s1*,*i*[,*s2*]**)**	Returns rightmost *i* characters of *s1*. If *i* is greater than `length(`*s1*`)` the result is p added with *s2*. As much of *s2* as is necessary will be used; if *s2* is too short it will be used multiple times. *s2* defaults to a space if it is omitted.
right(*l1*,*i*[,*l2*]**)**	Returns rightmost *i* elements of *l1*. If *i* is greater than `length(`*l1*`)` the result is padded with *l2*. As much of *l2* as is necessary will be used; if *l2* is too short it will be used multiple times. *l2* defaults to one empty list element if it is omitted.
role	Operate on `role` options.
set_channel(*s*,*c***)**	Modify the channel named by *s* with the channel options specified by the optlist *c*. The channel will be created if it doesn't already exist.

set_option(*s*[,*v*]**)**	Set option *s* (for options that take no value), or set option *s* to the string value *v*. An error will occur if the specified option name is invalid or matches multiple options.
set_options(*o***)**	Set zero or more options specified in optlist form. Option names and values are taken from optlist *o* in the obvious way. An error will occur if any of the specified option names are invalid or match multiple options.
set_statefile(*v*,*s***)**	Set the statefile variable *v* to the value *s*. The function returns `true` if the delete operation was succesful; `false` if statefile support is not enabled.
sign(*i***)**	Returns `-1` if *i* < `0`, `0` if *i* = `0`, `+1` if *i* > `0`.
sort(*l1*[,*i*[,*l2*]]**)**	Sorts the elements of *l1* to be in ascending order if *i* <> `0` and descending order if *i* = `0`. *i* defaults to `1` if it is omitted. If *l2* is present, its elements are shifted in the same way as elements in *l1* are shifted.
split(*s*[,*c*[,*i*]]**)**	Produces a list of elements consisting of pieces of *s* delineated by characters in *c*. If omitted, *c* defaults to a comma. If *i* is `0` or `1`, zero length elements are preserved; if *i* is `2`, they are not. If omitted, *i* defaults to `1`.
split(*l*[,*c*[,*i*]]**)**	Produces a list of elements consisting of pieces of elements of *l* delineated by characters in *c*. If omitted, *c* defaults to a comma. If *i* is `0`, boundaries between the original elements aren't preserved and zero length elements can be output; if *i* is `1`, boundaries are preserved and zero length elements can be output; if *i* is `2`, boundaries aren't preserved and zero length elements are omitted. If omitted, *i* defaults to `1`.
string(*e***)**	Converts *e* to a string. If *e* is already a string, it is returned unchanged. If *e* is an integer, it is converted to a string. If *e* is a list, the string that results from concatenating the elements of *e* is returned.
string(*l*,*s***)**	Converts the list *l* to a string, inserting the string *s* between each pair of elements of *l*. So for instance `string(["a","cd","e"],"01")` would return the string `"a01cd01e"`.
string(*i*[,*j*[,*k*]]**)**	Converts the integer *i* to a string, optionally padding with zeros (on the left) so that the length of the string is *j*, and optionally outputting the result in the radix specified by *k*. So for instance `string(15,8,2)` returns `00001111`.
strongrandom(*n***)**	Returns a string containing the specified number of bytes of random value.
translate(*s1*,*s2*,*s3***)**	Interprets the string *s1* as being in the character set specified by *s2* and returns a version translated into the character set specified by *s3*.
trim(*s*[,*c*]**)**	Returns *s* with any trailing characters found in *c* removed. *c* defaults to space and tab if omitted.
trim(*l*[,*c*]**)**	Returns list *l* with any trailing characters found in *c* removed from each element. *c* defaults to space and tab if omitted.
type(*e***)**	Returns "`integer`" if *e* evaluates to an integer, "`string`" if *e* evaluates to a string, and "`list`" if *e* evaluates to a list.
ucase(*e***)**	Converts any lower case characters in *e* to upper case. If *e* is a number it is converted to a string.

unset_alias(*s*,*v*)	New in 8.0.1.2. Remove the option or options specified in the string or list *v* from the alias named *s*. Note that this function is unsuitable for use on alias entries.
unset_channel(*s*,*v*)	Remove the option or options specified in the string or list *v* from the channel named *s*.
unset_option(*s*)	Removes option *s* from the configuration. An error will occur if the specified option name is invalid or matches multiple options.
validate_option(*s1*[,*s2*])	Validate the option named *s1*: that the option name is valid and, if the optional value *s2* is supplied, that the value is a valid value. Unlike `set_option`, `validate_option` does not set the option to the specified value: it merely checks validity.
warn(*s*)	Print warning text *s* to the administrator's terminal, and update the internal setting (which may be tested via `continue`) that a (an additional) warning has occurred.
write_file(*s1*,*s2*)	Writes the contents of string *s2* into the file named by *s1*. `msconfig` writes to the MTA configuration directory, `IMTA_TABLE:`, if *s1* is merely a file name with no patch. An error occurs if the file cannot be opened or written. Line feeds should be used as line separators in the string.
write_file(*s1*,*l*[,*s2*])	Writes the contents of the list *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. An error occurs if the file cannot be opened or written.
write_optlist(*o*)	Convert the optlist *o* 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 `write_file`.
yesno(*s1*[,*m*[,*s2*]])	Prompt at the recipe administrator's terminal with prompt string *s1*. A response of "`y`", "`Y`", "`t`", "`T`", or "`1`" will be accepted as meaning yes; a response of "`n`", "`N`", "`f`", "`F`", or "`0`" will be accepted as meaning no; these are the valid responses. *m* is an integer value to use as the default if the administrator merely returns (NULL response). *s2* is a warning string to output if the administrator's input was not in the valid set of responses.

4.7.1 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)
```

As of MS 8.0.2.1, the default value for an option setting can be determined with the **get_default** function. It returns the built-in default for the option. An empty string is returned if the option has no built-in default value. An error occurs if the specified option does not exist.

The **list_names**(*s*[,*n*]) function returns, as an optlist, a list of the options (and their values) whose names begin with the specified string *s*. The optional integer second argument, *n*, specifies whether or not to retain backslash characters (for quoting special characters) in the options' values; the default is 0 (false).

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 in either or two cases: if the specified option name is valid and the option is a no-value option which is set, or if the specified option name is valid for a valued option which is not set. (So note that get_option is not sufficient for checking whether a no-value option is set; instead use exists_option to check on a whether or not a no-value option is set.)

```
get_option("mm_debug")        -> 0       (mm_debug is set to 0)
get_option("os_debug")        -> ""      (os_debug, valued option, is not set)
get_option("slave_debug")     -> ""      (slave_debug, no-value, may be set)
exists_option("slave_debug") -> 0        (slave_debug not set)
get_option("*_debug")         -> <error> (multiple *_debug options)
```

The function **get_options**(*s*) returns an optlist containing the names and values of the options with name *s*. (So note that unlike get_option, the result returned by get_options has no ambiguity and differentiates between set no-value options *vs.* unset valued options.) Note that when both role and instance flavors of an option are set get_options returns only the instance flavor of the option.

The function **set_option**(*s*[,*v*]) sets the option named *s* for options that take no value, or sets the option names *s* to the value *v* for options that do take a value. The function **set_options**(*o*) sets the option-value pairs in the optlist *o*.

The **unset_option**(*s*) unsets (deletes) the option named by the string *s*. The **delete_options**(*l*) function deletes (unsets) the options named in the list *l*.

New in MS 8.0.2, recipes support a **resolve_option** function. This function accepts a single string argument which then undergoes the option resolution process. The result of that process is then returned as a bit-encoded integer. The bits are:

Table 4.3 resolve_option return bits

Bit	Value	Meaning
0	1	Set if name resolved, clear if name invalid
1	2	Set if the string specified a valid option name, clear otherwise (Note that an option string can specify the name of a group, in which case this bit will be clear but bit 0 will be set.)
2	4	Set if option has been deleted and no longer has any effect
3	8	Set if option is restricted
4	16	Set if option is targeted for use in instances only

The **validate_option**(*s1*[,*s2*]) function validates that the specified option name is valid, and that the specified value is valid. It does *not* actually set the option to the specified value. For instance:

```
validate_option("service:SMTP.enable","test")     -> 0
validate_option("service:SMTP.enable","0")        -> 1
```

The **instance** and **role** functions tell the recipe to set the specified flavor of option. The **default** function tells the recipe to set options according to the option's own preferred flavor (which note is `role` for all but a few options).

For enabling modification of restricted options, see also the `-restricted` switch to the `run` command, or the **restricted** operation.

4.7.1.1 Configuration group access

In Unified Configuration, some options may be set under a named group. So there are function to access/set/delete/*etc*. an entire such named group of options.

The **add_group**(*g*,*c*) function adds the group named (by the string value of) *g* with options specified by the optlist *c*; the function returns `1` if this operation was successful, or `0` if there was an error (such as in the case where the specified group name already existed). For example, the call:

```
add_group("dispatcher.service:SMTP_SPECIAL",
          ["enable", "1",
           "tcp_ports", "28225",
           "image", "IMTA_BIN:tcp_smtp_server",
           "parameter", "CHANNEL=tcp_special"
           "logfilename", "IMTA_LOG:tcp_special_server.log"])
```

adds a Dispatcher service named `SMTP_SPECIAL`, which will be a "special" SMTP server listening on port `28225` running the channel `tcp_special`.

The **append_group**(*s*,*c*) function appends to the group named (by the string value of) *s* the options specified by the optlist*c*; the function returns `1` if this operation was successful, or `0` if an error occurred. (The group will be created if it does not already exist; that is, `append_group` on a previously non-existent group is not an error.)

The **delete_group**(*s*) function deletes the group named (by the string value of) *s*.

The **exists_group**(*s*) function returns `0` if the group named (by the string value of) *s* exists, or the number of options set in the group if the group does exist.

The **get_group**(*s*) function returns the content of the group (the option names and values) as an optlist.

The **prepend_group**(*s*,*c*) function prepends to the group named (by the string value of) *s* the options specified by the optlist*c*; the function returns `1` if this operation was successful, or `0` if an error occurred. (The group will be created if it does not already exist; that is, `prepend_group` on a previously non-existent group is not an error.)

The **replace_group**(*s*,*c*) function replaces the group named (by the string value of) *s* with options specified by the optlist*c*; the function returns 1 if this operation was successful, or 0 if there was an error (such as in the case where the specified group name does not already exist).

4.7.2 System information

New in 8.0.1.2. The get_system_info function provides access to various pieces of static information about the system msconfig is running on. The function accepts a single case-insensitive string as its only argument specifying the piece of information to return.

In the following table, the Messaging Server version is taken to be "a.b.c.d" and the product build date is "yyyymmdd".

Table 4.4 get_system_info information items

Item	Return Type	Example Value	Description
ms_build_date	string	"yyyymmdd"	The date the current system was built from source.
ms_build_date_int	integer	yyyymmdd	The date the current system was built from source.
ms_version	string	"a.b.c.d"	Messaging Server version
ms_version_int	integer	aabbccddd	Messaging Server version, expressed as a single integer computed by the formula ((a * 100 + b) * 100 + c) * 1000 + d. This form is intended to be used in comparison operations.
ms_version_major	string	"a"	Messaging Server major version.
ms_version_major_int	integer	a	Messaging Server major version.
ms_version_minor	string	"b"	Messaging Server minor version.
ms_version_minor_int	integer	b	Messaging Server minor version.
ms_version_update	string	"c"	Messaging Server update version.
ms_version_update_int	integer	c	Messaging Server update version
ms_version_patch	string	"d"	Messaging Server patch number.
ms_version_patch_int	integer	d	Messaging Server patch number.
ms_version_build	string	""	Messaging Server build number.
ms_version_build_int	integer		Messaging Server build number.

4.7.3 msconfig information

New in 8.0.1.2. The get_msconfig_info function provides access to various pieces of information about how msconfig was invoked and its current status.

The following information items are currently supported:

Table 4.5 get_msconfig_info information items

Item	Return Type	Example Value	Description

noprompt	boolean	false	Returns `true` if `msconfig` was invoked with the -noprompt switch.
statefile_name	string	file.state	The name of the statefile specified with the -statefile switch.

4.7.4 Environment access

The recipe language can find out the values of environment variables, or find file path specifications using Messaging Server's basic environment variable values or (for the MTA) special symbolic names.

The **`getenv(`***`s`***`)`** function returns the value of the environment variable named *s*, or the empty string if no such environment variable is defined.

The **`get_path(`***`s`***`)`** function takes a string `"server"`, `"data"`, or `"config"` as argument, returning the file path specification of the SERVERROOT, DATAROOT, or CONFIGROOT, respectively.

The **`make_path(`***`s`***`)`** function will return a file path specification by converting any environment variable or special symbolic name in the string argument `s` into its path equivalent.

The **`get_option_modification_locations`** function will return a list of locations in the current recipe where a configuration modification occurred. This action also clears the internal list, so subsequent calls will only return any additional modifications that have occurred. An empty list is returned if the configuration has not been changed by the current recipe.

4.7.5 `msconfig` information operations

The **`argc`** function returns the number of additional arguments given to the `msconfig RUN` command.

The **`argv`**`(`*`n`*`)` function returns, as a string, the *n*th argument given to the `msconfig RUN` command. The value *n* must be between `1` and `argc` inclusive.

The **`description`**`(`*`d`*`)` function sets a string to display when the recipe is listed by the `msconfig DIRECTORY` command. Note that *d*must be a literal string enclosed in quotes; it cannot be an expression. When the recipe itself is executed, the **`description`** function does nothing other than return the value of its argument *d*.

The **`keywords`**`(`*`l`*`)` function specifies keywords to display when the recipe is listed by the `msconfig DIRECTORY` command.

4.7.6 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/"
```

For manipulating paths in file names, see also the `get_path` and `make_path` functions.

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 list *`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.

(New in MS 8.0.1.2) The **`delete_file(`***`s`***`)`** function deletes the file named by the string *`s`*, returning `1` if the delete operation was successful, or `0` if not.

Unless `-trusted` was specified for the `msconfig` invocation, the utility will check whether to perform a requested `delete_file` operation:

```
Allow recipe to delete file [Y, N, A]?
```

and if such prompting is disallowed (`-noprompt` specified), will not execute the delete operaiton and instead issue an error:

```
Delete_file not allowed to prompt for permission in -noprompt mode
```

4.7.7 Terminal I/O operations

The recipe language has a few terminal I/O primitives.

The **`print(`***`s`***`)`** operation prints a string to the terminal of the administrator executing the recipe.

The **`warn(`***`s`***`)`** operation prints a warning string to the terminal of the administrator executing the recipe, and updates the internal setting (which may be tested via **`continue`**) that a warning (an additional warning) has occurred.

New in MS 8.0.2.3, warn without any arguments returns the number of warnings that have been issued during the current run.

The **`error(`***`s`***`)`** operation issues a specified error string back to the administrator.

The **continue(***[s1[,s2]]***)** operation asks the administrator whether to continue after a warning. (Note that the prompt string argument *s1* and default response string argument *s2* are optional, with a default prompt string and empty response string, interpreted as false, used they are omitted. So merely **continue** is also valid syntax.)

The **yesno(***s1[,m[,s2]]***)** operation asks the administrator to respond with a yes or no response.

The **read(***s1[,s2[,s3]]***)** operation reads a string input from the terminal of the administrator executing the recipe. *s1* specifies the prompt to print on the terminal, *s2* specifies a default to return if no value is entered or `--noprompt` was specified when msconfig was invoked, and (new in 8.0.2.3) *s2* specifies the name of a statefile variable whose value is used as a default and updated with any value that's entered. The `--statefile` switch must be specified on the msconfig command line for *s2* to have any effect.

The **read_password** operation obtains a password string from the administrator executing the recipe.

If the `-restricted` switch was specified on a `run` command, the **restricted** operation merely returns a `1` unconditionally. But otherwise, the **restricted** operation prompts the administrator for whether or not to enable modification access to restricted options.

4.7.8 Statefile operations

The functions *exists_statefile*, *get_statefile*, *set_statefile*, and *delete_statefile* can be used to check the existence of, get, set, and delete statefile variables, respectively. Note that *exists_statefile* returns -1 and the other functions are no-ops if statefile suport is not enabled by specifying the `--statefile` switch on the msconfig command line.

4.7.9 Alias creation and manipulation operations

Messaging Server 8.0.1.2 now provides a set of recipe functions to create and manipulate aliases.

An MTA alias consists of a named set of option-value pairs, always containing one ore more `alias_entry` options which specify the alias expansion addresses. A single alias is represented in the recipe language using an optlist, and a number of functions are provided to access and manipulate aliases. All of these functions accept the unquoted name of the alias as the first argument. This name may be specified in any case and is converted to lower case.

Alias existence can be checked with **exists_alias**. A nonzero value is returned if the named alias is already part of the configuration; zero if it isn't.

The contents of an alias definition can be retrieved as an optlist using the **get_alias** function. For example, if the configuration has a `users-prefix@example.com` alias:

```
user-prefix@example.com: [prefix_text] Prefix text, user1@example.com, user2@example.com
```

The call `get_alias("user-prefix@exmaple.com")` will return an optlist:

```
["alias_prefix_text", "Prefix text",
 "alias_entry", "user1@example.com",
```

```
 "alias_entry", "user2@example.com"]
```

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

The **add_alias** function is used to add a new alias to the configuration. A second argument is required specifying the various alias options as an optlist. An optional third parameter can also be specified containing a list of alias entries; these are converted to `alias_entry` alias options. An error is returned if the alias already exists. For example, the call:

```
add_alias("list@example.com",
          ["envelope_from", "list-error@example.com"],
          ["list1@example.com, "list2@example.com])
```

adds this alias to the configuration:

```
list@example.com: [envelope_from] list-error@example.com, list1@example.com, list2@example.com
```

The **replace_alias** function is the same as `add_alias`, except that any alias with that name that already exists will be removed prior to the addition.

The **append_alias** function is also the same as `add_alias`, except that specified alias options and entries will be appended to any existing alias definition.

The **preend_alias** function is also the same as `add_alias`, except that specified alias options and entries will be preended to any existing alias definition.

The **delete_alias** function deletes the named channel from the configuration. No operation is performed if the alias doesn't exist.

The **unset_alias** function takes two arguments: The name of an existing alias and a list of alias options to delete from it. Note that unset_alias is not designed to work on alias entries.

4.7.10 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 channel definition 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 mustsaslserver smtp allowswitchchannel maytlsserver
tcp_tas-daemon
```

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

```
["official_host_name", "tcp_tas-daemon",
 "deliveryflags", "2",
 "mustsaslserver", "",
 "smtp", "",
 "allowswitchchannel", "",
 "maytlsserver", ""]
```

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 channel 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", "",
                        "randonmx", "",
                        "noswitchchannel", "",
                        "pool", "SMTP_POOL",
                        "smtp", ""]);
```

adds this channel to the configuration:

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

The fact that a channel is represented as an optlist makes it easy to add a channel based on an existing one:

```
add_channel("tcp_new",
            put_optlist(get_channel("tcp_local"),
                        "official_host_name", "tcp-new"));
```

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** function changes an existing channel. The second argument to **set_channel** 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 maysaslserver mx pool SMTP_POOL \
 saslswitchchannel tcp_auth single_sys smtp \
 allowswitchchannel maytlsserver
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 daemon router.example.com loopcheck master_debug \
 maysaslserver multiple nomx pool SMTP_POOL \
 saslswitchchannel tcp_auth single_sys smtp \
 allowswitchchannel maytlsserver
tcp_intranet-daemon
```

4.7.11 Rewrite rule creation and manipulation operations

MTA rewrite rules are represented in Unified Configuration a list of `rule` values under the `rewrite` group. Each such rule value has a pattern and a template, separated by white space. The recipe language supports a number of functions which operate on rewrite rules; for rewrite rule values, such functions use an optlist specifying pattern-template pairs. For instance, the call:

```
append_rewrites([".lmtp", "$E$F$U%$H.lmtp@lmtpcs-daemon",
                 ".lmtp", "$B$F$U%$H@$H@lmtpcs-daemon"])
```

adds rewrite rules:

```
msconfig> show rewrite * .lmtp*
role.rewrite.rule = .lmtp $E$F$U%$H.lmtp@lmtpcs-daemon
role.rewrite.rule = .lmtp $B$F$U%$H@$H@lmtpcs-daemon
```

Note how, in contrast to many other recipe language function uses of optlists, when it comes to the rewrite rule creation and manipulation functions, the optlist arguments used are not specifying option-value pairs, but rather are specifying (an ordered rule list of) pattern-template pairs.

The recipe language functions available specifically for creating or manipulating rewrite rules are:

- **append_rewrites**(*c*)
- **delete_rewrites**(*l*)
- **get_rewrites**[(*p*[,*t*])]
- **prepend_rewrites**(*c*)
- **replace_rewrites**(*c*)

4.7.12 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 addresse
  *|*|*|*|*                     $C$|INTERNAL_IP;$3|$Y$E
  *                             $NEXTERNAL
```

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

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

then 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
```

4.7.13 Deployment map operations

The `deploymap` recipe language function's semantics closely follow those of the DEPLOYMAP `msconfig` command.

All of the deploymap recipe and msconfig support was added in the 8.0.1.1.0 release of Messaging Server.

4.7.13.1 Add operations

```
deploymap :add :deployment d
```

Adds deployment *d* to the deployment. Adding a deployment that already exists is a no-op.

```
deploymap :add [:deployment d] :host h [:role r]
```

Adds host(s) *h* with role *r* to deployment *d* in the deployment map. *h* can be either a string or a list. An error occurs if any of the hosts already exist. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. A deployment with the default name of "`site-01`" will be created if no deployment exists in the deployment map. No role will be associated with the hosts if *r* is not specified.

```
deploymap :add [:deployment d] :host h :property p
```

Add property/properties *p* to host *h* in deployment *d*. Host *h* will be created if it does not already exist. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. A deployment with the default name of "`site-01`" will be created if no deployment exists in the deployment map.

```
deploymap :add [:deployment d] :host h :property p :role r
```

The host *h* is created in deployment *d* with role *r* and properties *p*. The host must not already exist. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. A deployment with the default name of "`site-01`" will be created if no deployment exists in the deployment map.

All add operations return the number of modifications that were made to the deployment map.

4.7.13.2 Create operations

```
deploymap :create
```

Create a new, empty deployment map. Note that if you write this out using `msconfig`'s `deploymap write` command you will delete all your existing entries!

4.7.13.3 Delete operations

```
deploymap :delete :deployment d
```

Deletes deployment *d* from the deployment map. Deleting a nonexistent deployment is a no-op.

```
deploymap :delete [:deployment d] :host h
```

Deletes host(s) *h* from deployment *d*. *h* can be either a string or a list. Deleting a nonexistent host is a no-op. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected.

```
deploymap :delete [:deployment d] :host h :role
```

Delete any role associated with host(s) *h* in deployment *d*. Deleting a nonexistent role is a no-op. An error occurs if host *h* does not exist. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected.

```
deploymap :delete [:deployment d] :host h :property p
```

Delete any properties for host *h* in deployment *d* that match any of the glob-style wildcard(s) *p*. Both *p* and *h* can be a string or a list. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected.

All delete operations return the number of modifications that were made to the deployment map.

4.7.13.4 Dump operation

```
deploymap :dump
```

Returns a string containing an outline of the entire deployment map.

4.7.13.5 List operations

```
deploymap :list :deployment [d]
```

Returns a list of all the deployments in the deployment map that match the pattern *d*. All deployments are returned if *d* is omitted.

```
deploymap :list [:deployment d] :host [h] [:online | :offline]
```

Returns a list of all of the hosts in deployment *d* which match the pattern glob-style pattern *h*. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. All hosts are returned if *h* is not specified.

```
deploymap :list [:deployment d] [:host h] :role [r] [:online | :offline]
```

Returns a list of all the unique roles of the hosts in deployment *d* matching the glob-style pattern *h*. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. *r* is an optional glob-style pattern used to filter the roles that are returned. A list of all the roles associated with all of the hosts is returned if `:host` *h* is not specified. Note that the expression:

```
string(deploymap :list :host h :role)
```

can be used to obtain the role of host *h* as a string.

```
deploymap :list [:deployment d] :host :role [r] [:online | :offline]
```

Returns an optlist containing a list of all the hosts in deployment *d* and their associated roles. Hosts that do not have roles are not returned. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. *r* is an optional glob-style pattern containing one or more wildcards or two or more non-wildcard strings used to filter the roles that are returned.

Note that a sublist `l` of the first elements in an optlist `t` is easily extracted with a loop of the form:

```
l = []; loop {exitif t == []; l = pop(t); pop(t);}
```

```
deploymap :list [:deployment d] :host :role r [:online | :offline]
```

Returns a list of all the hosts in deployment *d* with role *r*. *r* must not contain more than one element or any glob-style wildcard characters; if it does an optlist is returned as described above. The current deployment is used if `d` is not specified; if there is no current deployment the first deployment in the deployment map is selected.

```
deploymap :list [:deployment d] [:host h] [:role r]
          :property [p] [:online | :offline]
```

Returns a list of property values for hosts matching *h* in deployment *d* and with a role matching *r*. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. If `:host` *h* is not specified the properties of all hosts are returned. No role check is performed if *r* is not specified. *p* is an optional glob-style pattern used to filter the properties that are returned.

```
deploymap :list [:deployment d] :host [:role r] :property p [:online | :offline]
```

Returns a list of hosts in deployment *d* which have properties matching *p*, and optionally, a role matching *r*. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. *p* must be a string or list of glob-style patterns.

In all of these operations the `:online` and `:offline` parameters will limit hosts to those known to be online or offline, respectively.

4.7.13.6 Read operations

```
deploymap :read s
```

Reads a JSON-format deployment map from string *s*.

4.7.13.7 Rename operations

```
deploymap :rename n :deployment d
```

Renames deployment *d* to *n*. *d* must specify an existing deployment; *n* must be a nonempty utf-8 string.

```
deploymap [:deployment d] :rename n :host h
```

Renamed host *h* in deployment *d* to *n*. The current deployment is used if *d* is not specified; if there is no current deployment the first deployment in the deployment map is selected. Host *h* must already exist and *n* must be a valid domain name.

All rename operations return the number of modifications that were made to the deployment map.

4.7.13.8 Set operations

```
deploymap :set [:deployment d] :host h :role r
```

Sets the role for host(s) *h* in deployment *d* to *r*. *h* can be either a string or list. The specified hosts must already exist in the deployment. The current deployment is used if `d` is not specified; if there is no current deployment the first deployment in the deployment map is selected.

All set operations return the number of modifications that were made to the deployment map.

4.7.13.9 Write operations

```
deploymap :write
```

Returns the contents of the current deployment map as a JSON-formatted string. Note that no mechanism is provided in the recipe language to update the active deployment map; this can only be done at the `msconfig` level.

4.7.14 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
```

And in MS 8.0.1.3 or later:

```
o = delete_optlist(o, ["A","B"]); # Delete option A with value B from optlist o
```

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

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

4.7.15 LDAP operations

(New in MS 8.0.1) The recipe language can access LDAP. Several functions exist for such operations.

The **ldap_init**(*o*) function initializes the built in LDAP client. It must be called prior to using any of the other recipe language ldap_* functions.

By default, the LDAP client uses the current settings of the ugldaphost, ugldapbinddn, ugldapbindcred, ugldapport, and ugldapusessl base options when it initializes. The single argument *o* 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.

The **ldap_ldif**(*s*[,*f*]) function applies the LDIF contained in the string *s* to an LDAP directory. (Details of which LDAP directory -- and how to connect to it -- must previously be established via an ldap_init call; attempting to call ldap_ldif before ldap_init will result in an error.)

By default, any LDAP errors that occur attempting to perform the operation of applying the LDAP will be treated as a recipe warning, but see bit 4096 of the optional flags argument *f*. The optional flag argument *f* is a bit-encoded integer specifying flags; the currently defined flag bits are:

Table 4.6 ldap_ldif optional flag argument bits

Bit	Value	Usage
0	1	If set, continute processing after any error.
1	2	If set, treat "entry exists" on add as success.
2	4	If set, allow no-such-object if hint present.
12	4096	If set, treat any LDAP error that occurs as a recipe error.

The ldap_ldif function returns an integer count of the number of successful modifications performed.

(New in MS 8.0.1.1.0) The **ldap_search** function takes an optlist argument specifying at least the basedn for the search, and optionally also the attribute to return (attrs), the scope of the search, and a filter (filter) for the search. Valid values for the "scope" are: base, onelevel (or one), subtree (or sub). An optional second integer argument specifies the number of entries to return, and defaults to -1 (return all entries) if omitted. The function returns an optlist containing the matching attribute-value pairs.

4.7.16 Random value generation

The **strongrandom**(*n*) function returns the specified number of bytes of random value. This is a cryptographically strong generator.

The **random**(*n*) function returns a uniformly distributed random number between 0 and n-1. The linear congruential generator described in "Random Number Generators: Good Ones Are Hard To Find", S. Park and K. Miller, CACM 31 No. 10, pp. 1192-1201, October 1988 is used to generate these numbers. The sequence is initially seeded with the sytem time.

An explicit 32 bit integer seed for the **random** function can be specified by calling the **randomseed**(*n*) function. This may be useful for debugging purposes. A value of 0 will cause the sequence to be reinitialized from the system clock.

4.7.17 Call-out to routine in external library

The **call_user**(*s1*,*s2*,*e*[,*e2...*]) function lets recipes call out to routines in external libraries. The argument *s1* specifies the path to the external library (assumed to be in IMTA_LIB if a full path is not provided); the argument *s2* specifies the routine name; and argument(s) *e*, *e2*, *e3*, *etc.*, specify the arguments to provide to routine *s2*. The result of the routine call is returned as the result of call_user.

call_user performs a call of the routine named *s2* of the form:

```
s2(int argc, struct argv, char *reason, int *rlength);
```

This feature is intended for calling out to site-supplied external routines.

4.8 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.

4.9 Preprocessing Directives

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

Table 4.7 Preprocessor Directives

Directive	Description
`%define` *name* [*value*]	Define the preprocessor symbol *name*. A optional value can be specified but nothing presently makes use of such values.
`%elifdef` *name*	Combines `%else` and `%ifdef`.
`%elifndef` *name*	Combines `%else` and `%ifndef`.
`%else`	Invert the processing state of the preceding `%ifdef` or `%ifndef`
`%endif`	Terminates the innermost `%ifdef` or `%ifndef` processing block. The processing state reverts to that of the enclosing block if there is one.
`%ifdef` *name*	Only process subsequent material up to a matching `%else` or `%endif` if *name* is defined.
`%ifndef` *name*	Only process subsequent material up to a matching `%else` or `%endif` if *name* is not defined.
`%include` *filename*	Include the content of the file *filename* in the recipe at the point where the `%include` directive appears. If a path is not provided in the file specification a series of paths will be checked: (1) The paths to any nested include files, innermost first, (3) The path to the recipe file specified in the RUN command, (4) `CONFIGROOT/recipes/`, and `SERVERROOT/lib/recipes/`. Checking (4) and (5) is new in MS 8.0.1.2.0.
`%undef` *name*	Remove a previous definition of *name*.

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.

4.10 Random number generation

Messaging Server uses the special device /dev/urandom for direct generation of random numbers on all platforms. The direct uses of random numbers include:

- The recipe language's strongrandom function.
- The strongrandom function provided for use in system-level sieves.
- Generation of authentication nonce values.
- Generation of initialization vectors when encrypting store message files
- Generation of recall/tracking secrets for message tracking and recall
- Generation of an internal key used for password obfuscation while preserving the ability to perform comparisons in the msconfig differences command.

Note: Random numbers needed for SSL/TLS operations are generated by the underlying cryptographic libraries.

Contrary to popular belief, /dev/urandom provides a high quality cryptographically secure random number source on all modern versions of Linux and Solaris. And with the possible exception of Solaris SPARC, the inclusion of entropy obtained from the HRNG provided by all recent Intel CPUs eliminates any "low entropy" conditions on startup.

For the one remaining case of Solaris SPARC, anyone concerned about a lack of entropy on startup can implement the following two procedures that preserve the entropy in the entropy pool across reboots:

```
echo "Initializing random number generator..."
random_seed=/var/run/random-seed
# Load and then save some entropy from the pool
if [ -f $random_seed ]; then
  cat $random_seed >/dev/urandom
else
  touch $random_seed
fi
chmod 600 $random_seed
dd if=/dev/urandom of=$random_seed count=1 bs=512
```

This first procedure should be run as root at system startup. The second procedure is:

```
# Carry a random seed from shut-down to start-up
# Save some entropy from the entropy pool
echo "Saving random entropy..."
random_seed=/var/run/random-seed
touch $random_seed
chmod 600 $random_seed
dd if=/dev/urandom of=$random_seed count=1 bs=512
```

This second procedure should be run as root at system shutdown as well as periodically.

Finally, a Messaging Server-specific trick that can be used to provide more entropy when unified configuration is used is to hash the Messaging Server configuration file and use the result as a source of entropy. This can be done with a command of the general form:

```
openssl dgst -sha512 /var/opt/sun/comms/messaging64/config/config.xml >/dev/urandom
```

This provides significantly more entropy than expected because every the configuration generation utilities in Messaging Server tag each option value with a last modified time.

Chapter 5 Sieve filters

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. Besides using Sieve filter tests in normal expiration rules (expiration rules defined via either Message Store expirerule options or for greater flexibility and performance `store.expirerule` files), new in MS 7.0.5 the `imexpire` utility also supports invoking spam/virus filter packages to scan messages post-

delivery for spam or virueses, integrating application of the verdicts from such spam/virus filter package via Sieve filter scriptlets.

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. As of MS 7.0, the MTA supports the Sieve "`ihave`" extension, which allows Sieve scripts to test which extensions are available. 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, grouped into:

- Control structures,
- Actions,
- Tests,
- Functions,
- Values,
- ADDRESS-PARTS,
- BODY-TRANSFORMS,
- COMPARATORS,
- DATE-PARTS,
- ENVELOPE-PARTS,
- ENVIRONMENT-ITEMS,
- MATCH-TYPES, and
- MIMEOPTS.

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 5228			Block of commands
`error` *string*	RFC 5463		`require "ihave";`	Terminate Sieve script with runtime error
`foreverypart` [`:processnestedmessages` / `:retainnestedmessages`] [`:name` *string*] *command-block*				
	RFC 5703		`require "foreverypart";`	(New in MS 8.0) Loop through the MIME parts of a message
`:name`	RFC 5703			Specify a name for this `foreverypart` loop for reference in enclosed `break` or `continue` statements
`:processnestedmessages`	Private			(New in 8.1) Process the inner parts of any nested messages (default)
`:retainnestedmessages`	Private			(New in 8.1) Treat nested messages as leaf parts
`break` [`:name` *string*]	RFC 5703			(New in MS 8.0) Break out of a `foreverypart` loop
`:name`	RFC 5703			Terminate closest enclosing loop having specified name

`continue` [`:name` *string*]	Private			(New in MS 8.0) Pass control to the bottom of the `foreverypart` loop
`:name`	RFC 5703			(New in MS 8.0) Pass control to the bottom of the closest enclosing `foreverypart` loop having specified name
`loop` [`exitif` *expression*]* [`nextif` *expression*]* *command-block*				
	Private	system-level		General loop
`exitif`	Private	system-level		Exit a `loop` structure
`exitif`	Private	system-level		(New in 8.0.1.3) Proceed to the next iteration of the `loop` structure
`if` *test command-block* [`elsif` *test command-block*]* [`else` *command-block*]				
	RFC 5228			Branch-on-condition control structure
`elsif` *test command-block*	RFC 5228			Next case of branch-on-condition structure
`else` *test command-block*	RFC 5228			Final case of branch-on-condition structure
`require` *capability-list*	RFC 5228	`strict_require` MTA option		Declare that a Sieve script may use the named extension(s)
`stop`	RFC 5228			End processing
`sub` *name*[(*variables-list*)] `{...}`				
	Private	system-level		(New in MS 8.0) Define a subroutine
`my` *var-name* [`:=` *value*]	Private			(New in MS 8.0) Declare a variable is local
`return` *value*	Private			(New in MS 8.0) Exit subroutine, returning *value*
Actions				
`addconversiontag` *string-or-list*				
	Private	system-level		System-level Sieve action to add the specified conversion tag(s) to the message
`addflag` [*variable-name*] *list-of-flags*				
	RFC 5232	(`max_variables` MTA option, if using optional variable argument)	`require "imap4flags";` (or `require ["imap4flags", "variables"];` if using optional variable argument)	Add the specified IMAP flag(s) to the message
`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
`addsuffix` *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 imsimta test -expression -mm -debug=2 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 "editheader";	Delete the specified header line; (note that per RFC 5293, attempts to delete Received: or Auto-submitted: header lines will be ignored). As of MS 8.1.0.1, glob-style wildcards may be used in the field name.
: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 5228, 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", "foreverypart", "variables"];	(New in MS 8.0) Store extracted message text into a variable
:first	RFC 5703			Extract only the first N characters of the current MIME body part into a variable
:encodeurl	RFC 5435			In a variablesset command or an extracttext 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 variablesset command or extracttext 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 *folder-name*				
	RFC 5228	max_fileintos MTA option, fileinto and flagtransfer channel options	require "fileinto";	Deliver into specified folder
:copy	RFC 3894		require "copy";	Modifies the fileinto and redirect actions so that the action is performed without affecting Sieve's "implicit keep"
:flags	RFC 5232		require "imap4flags";	Used with keep or fileinto 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 postmaster, 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 `.HELD` file in the MTA queue area; ignored (no error) in user-level Sieve scripts
`importanceadjust` *value-string*				
	Private			Set or adjust a message's importance
`jettison` [*log-string*]				
	Private	`filter_jettison` and `log_filter` MTA options, `logging` channel option	`require "jettison";`	Non-overridable discard; a *log-string* may be specified to be included in the `log_filter` field in MTA message transaction log entries
`keep`	RFC 5228			Deliver message "normally"
`:flags`	RFC 5232		`require "imap4flags";`	Used with `keep` or `fileinto` 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` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] [`:timeout` *timeout-number*] *key-string value-string*				
	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` *value* [`:host` *host-str*] [`:tableprefix` *prefix-str*] [`:duplicate`] [`:quotatimeout` *quotatimeout-num* [`:penalize`]] [`:timeout` *timeout-num*] *key-str*				
	Private			Decrement the entry with the specified key by the specified adjustment value. The entry must be an unsigned integer written in string form or a throttle entry.
`:adjustup` *value* [`:host` *host-str*] [`:tableprefix` *prefix-str*] [`:duplicate`] [`:quotatimeout` *quotatimeout-num* [`:penalize`]] [`:timeout` *timeout-num*] *key-str*				
	Private			Increment the entry with the specified key by the specified adjustment value. The entry must be an unsigned integer written in string form or a throttle entry.
`:append` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] *key-string value*				
	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` *host-string*] [`:tableprefix` *prefix-string*] *key-string*				
	Private			Fetches the value of the entry with the specified key, or return an empty string if the entry doesn't exist. For throttle or reservation entries the current hit or reservation count, respectively, is returned.
`:prepend` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] *key-string value-string*				
	Private			Prepend the specified value to the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.)
`:remove` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] [`:lockout` *lockout-numeric*] *key-string*				
	Private			Remove the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.)
`:replace` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string value-string*				
	Private			Update the value and timeout of an (existing) entry. (Returns TRUE if the operation is successful, FALSE if it is not.)
`:store` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string value-string*				
	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.)
`metermaid`	Private	`enable_sieve_metermaid` 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
`nonotify`	Private	system-level		System-level Sieve action to suppress all applications of either form of `notify` action
`notify :method "email"` [`:id` *string*] `:options` *recipient-addr* [*subject-str*] [< `:days` / `:hours` / `:seconds` > *num*] [`:low` / **`:normal`** / `:high`] [`:mime`] [`:message` *message-str*]				
	draft-martin-sieve-notify-01	`max_notifys` MTA option	`require "notify";`	Generate a new message, typically some form of notification message
`:high`	`draft-...-01`			Urgent priority
`:id`	`draft-...-01`			
`:low`	`draft-...-01`			Non-urgent priority
`:method`	`draft-...-01`			The type of notification to generate: only `"email"` is supported
`:normal`	`draft-...-01`			Normal priority

:options	draft-...-01			:method-specific options; for the "email" method, the option value is the recipient email address to which to send the generated notification
:echo	Private			
:mime	Private			The specified message body is a MIME entity, including MIME headers as well as content
:days	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of days
:hours	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of hours
:seconds	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of seconds
notify [:from *string*] [:importance < "1" / "2" / "3" >] [:options *string-list*] [:mime] [:message *string*] *method-string*				
	RFC 5435	max_notifys, notify_ignore_errors MTA options	require "enotify";	Generate a new message, typically some form of notification message
:fcc	draft-ietf-extra-sieve-fcc-02.txt			Specify mailbox to receive copy of any generated notification
:from	RFC 5435			Specify author (From address) of notification
:importance	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)
:options	RFC 5435			
:message	RFC 5435			
:echo	Private			
:mime	Private			The specified message body is a MIME entity, including MIME headers as well as content
:days	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of days
:hours	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of hours
:seconds	Private	notify_minimum_timeout, notify_maximum_timeout, and notify_timeout_default MTA options		Specify duplicate notification timeout in units of seconds
novacation	Private	system-level		System-level Sieve action to suppress all applications of the vacation action
override	Private		require "override";	(New in MS 8.0) Mark this Sieve script as determining the disposition of a message
redirect [MODIFIERS] *address*				
	RFC 5228	max_redirects, max_redirect_addresses MTA options		"Forward" message to specified recipient address(es)
:copy	RFC 3894		require "copy";	Modifies the fileinto and redirect actions so that the action is performed without affecting Sieve's "implicit keep"
:keepmailfrom	Private			Retain original envelope FROM address on redirect; :keepmailfrom is the default unless :notify is specified in which case the default switches to :resetmailfrom
:list	RFC 6134	SIEVE_EXTLISTS mapping table, max_redirect_addresses MTA option	require "extlists";	Externally stored list of addresses to which to forward with redirect action
:noresent	Private	sieve_redirect_add_resent MTA option		Do not add Resent-*: header lines upon redirect
:notify	RFC 6009		require "redirect-dsn";	Set message's NOTIFY parameter on redirect
:resent	Private	sieve_redirect_add_resent MTA option		Add Resent-*: header lines upon redirect
:resetmailfrom	Private			Reset envelope FROM address to Sieve owner on redirect
:ret	RFC 6009		require "redirect-dsn";	Set message's NOTIFY RET (return-of-content) parameter on redirect

`redis`	Private	`redis.hostlist`, `redis.port`, `redis.servicename`, and `enable_sieve_redis` MTA options		(New in MS 8.0.2.3) Used as an action to manipulate data via the redis protocol; used as a test to access data via the redis protocol
`:add` [`:set`] [`:sortedset`] [`:score` *score*] [`:tableprefix` *prefix-string*] [`:duplicate`] [`:timeout` *timeout-number*] *key-string value-string*				
	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. When a set or sorted set is specified the entry is added to the set; if the set doesn't exist it will be created with the single specified. A score of 0 is used if no score is specified for a sorted set entry.
`:adjustdown` *value* [`:tableprefix` *prefix-str*] [`:duplicate`] [`:quotatimeout` *quotatimeout-num* [`:penalize`]] [`:timeout` *timeout-num*] [`:sortedset`] *key-str* [*set-member*]				
	Private			Decrement the entry with the specified key by the specified adjustment value. The entry must be an integer written in string form, a throttle entry, or a sorted set. Note that *set-member* can only be specified if `:sortedset` is also specified. The adjusted value is returned.
`:adjustup` *value* [`:tableprefix` *prefix-str*] [`:duplicate`] [`:quotatimeout` *quotatimeout-num* [`:penalize`]] [`:timeout` *timeout-num*] [`:sortedset`] *key-str* [*set-member*]				
	Private			Increment the entry with the specified key by the specified adjustment value. The entry must be an integer written in string form, a throttle entry, or a sorted set. Note that *set-member* can only be specified if `:sortedset` is also specified. The adjusted value is returned.
`:append` [`:tableprefix` *prefix-string*] [`:duplicate`] *key-string value*				
	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` [`:set`] [`:sortedset`] [`:withscores`] [`:tableprefix` *prefix-string*] *key-string*				
	Private			Fetches the value of the entry with the specified key, or return an empty string if the entry doesn't exist. For throttle or reservation entries the current hit or reservation count, respectively, is returned. Set and sorted set members are returned as lists. `:withscores` in conjunction with `:sortedset` causes set members and their associated scores to be returned as pairs in the list.
`:remove` [`:tableprefix` *prefix-string*] [`:duplicate`] *key-string*				
	Private			Remove the entry with the specified key. (Returns TRUE if the operation is successful, FALSE if it is not.) Note that `:remove` works regardless of data type; there is no need to specify additional parameters when dealing with sets or sorted sets.
`:replace` [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string value-string*				
	Private			Update the value and timeout of an (existing) entry. (Returns TRUE if the operation is successful, FALSE if it is not.)
`:store` [`:set`] [`:sortedset`] [`:score` *score*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string value-string*				
	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.) When a set or sorted set is specified the entry is added to the set; if the set doesn't exist it will be created with the single specified. A score of 0 is used if no score is specified for a sorted set entry.
`refuse` *string*				
	Private	++	`require "refuse";`	Refuse message, at SMTP level if possible, falling back to generating MDN
`reject` *string*				
	RFC 5429		`require "reject";`	Refuse message
`removeconversiontag` *string-or-list*				
	Private	system-level		System-level Sieve action to remove the specified conversion tag(s) from the message
`removeflag` [*variable-name*] *list-of-flags*				
	RFC 5232	(`max_variables` MTA option, if using optional variable argument)	`require "imap4flags";` (or `require ["imap4flags", "variables"];` if using optional variable argument)	Remove the specified IMAP flag(s) from the message

`replaceheader` [`:index` *value* [`:last`]] [`:newname` *header-field-name-str*] [`:newvalue` *string*] [COMPARATOR] [MATCH-TYPE] *header-field-name-str* [*value-patterns-list*]				
	draft-degener-sieve-editheader-00		`require "editheader";`	Replace a header line
`:index`	`draft-...-00`			Separate though equivalent to the RFC 5260 "`:index`" 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
`:last`	`draft-...-00`			For the "`:index`" modifier, count backwards from the end of the message header
`:newname`	`draft-...-00`			Change the name of all matching header fields to the specified new name
`:newvalue`	`draft-...-00`			Change the value of all matching header fields to the specified new value
`set` [MODIFIERS] *variable-name string-value*				
	RFC 5229	`max_variables` MTA option	`require "variables";`	Set a variable to a value
`:encodeurl`	RFC 5435			In a `variables set` command, perform percent-encoding (as per RFC 3986) of the string value
`:length`	RFC 5229			Decimal number of characters in the value 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 string left unchanged
`:quoteregex`	Private			In a `variables set` 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 string left unchanged
`setconversiontag` *string-or-list*				
	Private	system-level		System-level Sieve action to set the message to have exactly the specified conversion tag(s)
`setenvelopefrom` *address-string-or-list*				
	Private	system-level		(New in MS 8.0) System-level Sieve action to override a message's original envelope From address
`setflag` [*variable-name*] *list-of-flags*				
	RFC 5232	(`max_variables` MTA option if using variables; with LMTP delivery, see also the `flagtransfer` channel option)	`require "imap4flags";` (or `require ["imap4flags", "variables"];` if using variables)	Set the message to have exactly the specified IMAP flag(s)
`setmtpriority` *integer-or-string*				
	Private	system-level		(New in MS 8.0) System-level Sieve action to set a message's MT-PRIORITY
`setnotify` *string-or-list*				
	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` *string-or-numeric-priority-value*				
	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` *numeric-value-string*				
	Private			Set a message's "spam level"
`transactionlog` *string-or-list*				

	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` [`:days` *number*] [`:subject` *string*] [`:from` *address*] [`:addresses` *string-list*] [`:mime`] [`:handle` *string*] *reason-string*				
	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)
`:addresses`	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
`:days`	RFC 5230	`autoreply_timeout_default`, `vacation_minimum_timeout`, and `vacation_maximum_timeout` MTA options		Specify timeout in days
`:echo`	Private			Modifier on `vacation` to produce a "processed" MDN response
`:fcc`	draft-ietf-extra-sieve-fcc-02.txt			Specify mailbox to receive copy of any generated vacation notice
`:from`	RFC 5230			Address to use in the From: header line of the generated vacation message
`:handle`	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)
`:headers`	Private			Modifier on `vacation` to produce a response containing header lines, rather than the default MDN called for by the standard
`:hours`	Private	`autoreply_timeout_default`, `vacation_minimum_timeout`, and `vacation_maximum_timeout` MTA options		Specify the `vacation` action's autoresponse period in hours, rather than the normal days
`:mime`	RFC 5230			The *reason-string* is a MIME entity, including MIME headers as well as content
`:noaddresses`	Private			Modifier on `vacation` 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
`:noheadercheck`	Private			(MS 8.0.2.3) Modifier on `vacation` to suppress the MTA'S normal requirement (as per RFC 5230, Section 4.6) to only respond if various headers, e.g., List-ID:, are not present in the original message
`:reply`	Private			Modifier on `vacation` to produce a pure reply, containing only the reply text, rather than the default MDN called for by the standard
`:seconds`	RFC 6131	`autoreply_timeout_default`, `vacation_minimum_timeout`, and `vacation_maximum_timeout` MTA options	`require "vacation-seconds";`	Specify `vacation` action's autoresponse period in seconds, rather than the normal days
`:subject`	RFC 5230			Text to use in Subject: header line of generated vacation message
`virusset` *numeric-value-string*				
	Private			Set a message's "virus level"
`warn` *string-or-list*				
	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] *header-list value-list*				
	RFC 5228			Match Internet address in structured headers
`:aindex`	Private			Match against Nth address on header line
`:index`	RFC 5260		`require "index";`	Match against Nth named header line for `address`, `header`, or `date` tests
`:last`	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
`:anychild`	RFC 5703		`require "mime";`	Modify `:mime` to look inside any nested parts
`:raw`	Private analogue of RFC 5173			(New in MS 7.0.5) Do not MIME decode any RFC 2047 MIME encoded-words
`:text`	Private analogue of RFC 5173			(New in MS 7.0.5) Perform MIME decoding of any RFC 2047 MIME encoded-words
`allof` *test-list*	RFC 5228			Logical AND of tests; *test-list* has the form `(`*test* [`,` *test*]*`)`
`anyof` *test-list*	RFC 5228			Logical OR of tests
`body` [COMPARATOR] [MATCH-TYPE] [BODY-TRANSFORM] *value-list*				
	RFC 5173	`enable_sieve_body` MTA option	`require "body";`	Match content in the body of an email message; (only supports `:contains` and `:is`MATCH-TYPES, and a limited set of COMPARATORS)
`currentdate` [`:zone` *time-zone*] [COMPARATOR] [MATCH-TYPE] *date-part value-list*				
	RFC 5260		`require "date";`	Compare against current date-time
`:zone`	RFC 5260		`require "date";`	Specify a time zone offset to which to shift the current date-time prior to testing
`date` [`:zone` *time-zone* / `:originalzone`] [COMPARATOR] [MATCH-TYPE] *header-name date-part value-list*				
	RFC 5260		`require "date";`	Match against date from a header line
`:originalzone`	RFC 5260		`require "date";`	Test retaining the time zone offset of the original date time
`:zone`	RFC 5260		`require "date";`	Specify a time zone offset to which to shift the date-time prior to testing
`duplicate` [`:handle` *string*] [`:header` *header-name* / `:uniqueid` *value*] [`:seconds` *timeout*] [`:last`]				
	`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
`:handle`	`draft-bosch-sieve-duplicate-09`		`require "duplicate";`	Distinguish, via handle name, this duplicate test from other duplicate tests
`:header`	`draft-bosch-sieve-duplicate-09`		`require "duplicate";`	Use as unique ID (for duplicate detection) the content of the specified header field
`:last`	`draft-bosch-sieve-duplicate-09`	`duplicate_timeout_default`	`require "duplicate";`	Modify `:seconds` interpretation (or if `:seconds` 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
`:mime`	Private		`require "mime";`	
`:anychild`	Private		`require "mime";`	
`:seconds`	`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
`:uniqueid`	`draft-bosch-sieve-duplicate-09`		`require "duplicate";`	Use the specified value as the unique ID (for duplicate detection)
`envelope` [COMPARATOR] [ADDRESS-PART] [MATCH-TYPE] *envelope-part-list value-list*				
	RFC 5228		`require "envelope";`	Match SMTP envelope address
`environment` [COMPARATOR] [MATCH-TYPE] *environment-item value-list*				
	RFC 5183		`require "environment";`	Match against operating environment information
`exists` *header-list value-list*				
	RFC 5228			Test whether specified header(s) are present in a message
`:list`	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 `currentdate`, `date`, `environment`, `hasflag`, `spamtest`, `string`, and `virustest`, and on `deleteheader` and `replaceheader` actions
`:mime` [`:anychild`] [MIMEOPTS]				
	RFC 5703		`require "mime";`	Alter `address`, `exists`, or `header` tests to check aspects of structured MIME header lines
`:anychild`	RFC 5703		`require "mime";`	Modify `:mime` to look inside any nested parts
`:regex`	Private enhancement	`enable_sieve_regex` MTA option	`require "regex";`	Regex match type

	of `draft-murchison-sieve-regex-08`			
`false`	RFC 5228			Always evaluate to FALSE
`hasflag` [MATCH-TYPE] [COMPARATOR] [*variable-list*] *list-of-flags*				
	RFC 5232	(`max_variables` MTA option if using optional variable argument)	`require "imap4flags";` (or `require ["imap4flags", "variables"];` if using optional variable argument)	Test whether the message has the specified IMAP flag(s)
`:list`	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 `currentdate`, `date`, `environment`, `hasflag`, `spamtest`, `string`, and `virustest`, and on `deleteheader` and `replaceheader` actions
`header` [COMPARATOR] [MATCH-TYPE] *header-list value-list*				
	RFC 5228	`defer_header_addition` and (new in MS 8.0) `sieve_received` MTA options		Match header
`:mime` [`:anychild`] [MIMEOPTS]				
	RFC 5703		`require "mime";`	Alter `address`, `exists`, or `header` tests to check aspects of structured MIME header lines
`:anychild`	RFC 5703		`require "mime";`	Modify `:mime` to look inside any nested parts
`:raw`	Private analogue of RFC 5173			(New in MS 7.0.5) Do not MIME decode any RFC 2047 MIME encoded-words
`:text`	Private analogue of RFC 5173			(New in MS 7.0.5) Perform MIME decoding of any RFC 2047 MIME encoded-words
`ihave` *capability-list*				
	RFC 5463		`require "ihave";`	Test which Sieve extensions are available
`importancetest` [COMPARATOR] [MATCH-TYPE] *value*				
	Private			Test a message's importance
`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
`memcache :adjustdown` *value* [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] [`:quotatimeout` *quotatimeout-numeric* [`:penalize`]] [`:timeout` *timeout-value*] *key-string* [*test-string*]				
	Private			For the entry with the specified key, adjust the value down as specified and compare that adjusted value against the specified *test-string* value, which same *test-string* then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to '`:value "lt"`' and '`comparator "i;ascii-numeric"`', respectively.
`memcache :adjustup` *value* [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] [`:quotatimeout` *quotatimeout-numeric* [`:penalize`]] [`:timeout` *timeout-value*] *key-string* [*test-string*]				
	Private			For the entry with the specified key, adjust the value up as specified and compare that adjusted value against the specified *test-string* value, which same *test-string* then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to '`:value "gt"`' and '`comparator "i;ascii-numeric"`', respectively.
`memcache :fetch` [`:host` *host-string*] [`:tableprefix` *prefix-string*] [MATCH-TYPE] [COMPARATOR] *key-string* [*test-string*]				
	Private			For the entry with the specified key, compare its current value against the *test-string*, setting the value to *test-string*. The test always fails if the entry does not exist. MATCH-TYPE and COMPARATOR default to `:is` and `"i;ascii-casemap"`, respectively.
`memcache :release` [`:duplicate`] [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string*				
	Private			Implement a generic reservation capability. :release releases a previous reservation made with :reserve. The test returns TRUE if the reservation if a reservation is successfully released, FALSE otherwise.
`memcache :reserve :quota` *quota-numeric* [`:duplicate`] [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string*				
	Private			Implement a generic reservation capability. :reserve attempts to take out a reservation for the specified *key-string*. *quota-numeric* specifies the maximum number of reservations that are allowed. The test returns TRUE if the reservation if the reservation is taken out successfully, FALSE otherwise.

				Multiple reservations may be taken out by multiple :reserve operations. Note that each :reserve should be matched by a subsequent :release.
`memcache :throttle :quota` *quota-numeric* [`:duplicate`] `:quotetimeout` *quotatimeout-numeric* [`:limit`] [`:penalize`] [`:host` *host-string*] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] [MATCH-TYPE] [COMPARATOR] [`:adjustup` *adjustment-value*] [`:adjustdown` *adjustment-value*] *key-string* [*test-string*]				
	Private			Implement the 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 *test-string* 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 *test-string* 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`.
`metermaid`	Private	`enable_sieve_metermaid` 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
`:adjustdown` *adjustment-value* [`:host` *host-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			Decrement the entry with the specified key in the specified table by the specified adjustment value.
`:adjustup` *adjustment-value* [`:host` *host-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			Increment the entry with the specified key in the specified table by the specified adjustment value.
`:fetch` [`:host` *host-string*] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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 *test-string* 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.
`:greylisting` [`:host` *host-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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 *test-string* is specified (MATCH-TYPE and COMPARATOR default to '`:is`' and '`:comparator "i;ascii-casemap"`', respectively), then increment the throttle and then apply the Sieve test to the value.
`:remove` [`:host` *host-string*] [`:duplicate`] *table-string key-string*				
	Private			Remove the entry with the specified key from the specified table. Returns TRUE if the operation is successful; FALSE if it is not.
`:store` [`:host` *host-string*] *table-string key-string value-string*				
	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.
`:throttle` [`:host`: *host-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string*				
	Private			MeterMaid throttle functionality. If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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"`'.)
`not` *test*	RFC 5228			Reverse value of test argument
`notify_method_capability` [COMPARATOR] [MATCH-TYPE] *notification-uri notification-capability value-list*				
	RFC 5435		`require "enotify";`	Test whether a capability of a method of notification is available to be performed as desired
`:list`	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 `currentdate`, `date`, `environment`, `hasflag`, `notify_method_capability`, `spamtest`, `string`, and `virustest`, and on `deleteheader` and `replaceheader` actions

`redis`	Private	`redis.hostlist`, `redis.port`, `redis.servicename`, and `enable_sieve_redis` MTA options		(New in MS 8.0.2.3) Used as an action to manipulate data via the Redis protocol; used as a test to access data via the Redis protocol
`redis :adjustdown` *value* [`:tableprefix` *prefix-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] [`:quotatimeout` *quotatimeout-numeric* [`:penalize`]] [`:timeout` *timeout-value*] [`:sortedset`] *key-string* [*test-string*]				
	Private			For the entry with the specified key, adjust the value down as specified and compare that adjusted value against the specified *test-string* value, which same *test-string* then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to '`:value "lt"`' and '`comparator "i;ascii-numeric"`', respectively.
`redis :adjustup` *value* [`:tableprefix` *prefix-string*] [`:duplicate`] [MATCH-TYPE] [COMPARATOR] [`:quotatimeout` *quotatimeout-numeric* [`:penalize`]] [`:timeout` *timeout-value*] *key-string* [*test-string*]				
	Private			For the entry with the specified key, adjust the value up as specified and compare that adjusted value against the specified *test-string* value, which same *test-string* then becomes the entry's new value. MATCH-TYPE and COMPARATOR default to '`:value "gt"`' and '`comparator "i;ascii-numeric"`', respectively.
`redis :expire` *timeout-value* [`:duplicate`] [`:tableprefix` *prefix-string*] *key-string*				
	Private			New in MS 8.1.0.6. Sets the expiration time in seconds for the specified entry. A time of 0 will cause the entry to be be deleted immediately. Negative values will cause the entry to be preserved indefinitely.
`redis :fetch` [`:tableprefix` *prefix-string*] [MATCH-TYPE] [COMPARATOR] *key-string* [*test-string*]				
	Private			For the entry with the specified key, compare its current value against the *test-string*, setting the value to *test-string*. The test always fails if the entry does not exist. MATCH-TYPE and COMPARATOR default to `:is` and `"i;ascii-casemap"`, respectively.
`redis :release` [`:duplicate`] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string*				
	Private			Implements a generic reservation capability. :release releases a previous reservation made with :reserve. The test returns TRUE if the reservation if a reservation is successfully released, FALSE otherwise.
`redis :reserve :quota` *quota-numeric* [`:duplicate`] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] *key-string*				
	Private			Implements a generic reservation capability. :reserve attempts to take out a reservation for the specified *key-string*. *quota-numeric* specifies the maximum number of reservations that are allowed. The test returns TRUE if the reservation if the reservation is taken out successfully, FALSE otherwise. Multiple reservations may be taken out by multiple :reserve operations. Note that each :reserve should be matched by a subsequent :release.
`redis :throttle :quota` *quota-numeric* [`:duplicate`] `:quotetimeout` *quotatimeout-numeric* [`:limit`] [`:penalize`] [`:tableprefix` *prefix-string*] [`:timeout` *timeout-value*] [MATCH-TYPE] [COMPARATOR] [`:adjustup` *adjustment-value*] [`:adjustdown` *adjustment-value*] *key-string* [*test-string*]				
	Private			Implements the 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 *test-string* 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 *test-string* 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`.
`metermaid`	Private	`enable_sieve_metermaid` 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
`:adjustdown` *adjustment-value* [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			Decrement the entry with the specified key in the specified table by the specified adjustment value.
`:adjustup` *adjustment-value* [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			Increment the entry with the specified key in the specified table by the specified adjustment value.
`:fetch` [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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 *test-string* 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.
`:greylisting` [`:duplicate`] [MATCH-TYPE] [COMPARATOR] *table-string key-string* [*test-string*]				
	Private			If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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 *test-string* is specified (MATCH-TYPE and COMPARATOR default to ' `:is`' and ' `:comparator "i;ascii-casemap"`', respectively), then increment the throttle and then apply the Sieve test to the value.
`:remove` [`:duplicate`] *table-string key-string*				
	Private			Remove the entry with the specified key from the specified table. Returns TRUE if the operation is successful; FALSE if it is not.
`:store` *table-string key-string value-string*				
	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.
`:throttle` [MATCH-TYPE] [COMPARATOR] *table-string key-string*				
	Private			MeterMaid throttle functionality. If none of MATCH-TYPE, COMPARATOR, nor *test-string* 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"`'.)
`size` < `:over` / `:under` > *limit-value*				
	RFC 5228			Test size of message
`:over`	RFC 5228			Match if message size is over specified value
`:under`	RFC 5228			Match if message size is under specified value
`spamtest` [COMPARATOR] [MATCH-TYPE] *value*				
	RFC 3685		`require "spamtest";` or `require "spamtestplus";`	Test a message's "spam level"
`:list`	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 `currentdate`, `date`, `environment`, `hasflag`, `notify_method_capability`, `spamtest`, `string`, and `virustest`, and on `deleteheader` and `replaceheader` actions
`:percent`	RFC 5235		`require "spamtestplus";`	Test a message's "spam level" as a percentage value
`string` [MATCH-TYPE] [COMPARATOR] *source-string-list value-list*				
	RFC 5229	`max_sieve_string_size` MTA option	`require "variables";`	Test the (string) value of variable(s)
`:list`	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 `currentdate`, `date`, `environment`, `hasflag`, `spamtest`, `string`, and `virustest`, and on `deleteheader` and `replaceheader` actions
`true`	RFC 5228			Always evaluate to TRUE
`valid_ext_list` *extlist-names-list*				
	RFC 6134		`require "extlists";`	Test whether an external list specification is valid (both syntactically and semantically)
`valid_notify_method` *notification-uris-list*				
	RFC 5435		`require "enotify";`	Test whether notification methods are supported and syntactically valid
`virustest` [COMPARATOR] [MATCH-TYPE] *value*				
	RFC 3685		`require "virustest";`	Test a message's "virus level"
`:count`	RFC 5231		`require "relational";`	Match type for numeric comparison on number of occurrences (as of MS 8.0.1.3, :count may also be used as a modifier to other match types in header and address tests to count and test the number of matches that occurred)

:list	RFC 6134	SIEVE_EXTLISTS mapping table	require "extlists";	Match against externally stored data in address, currentdate, date, envelope, exists, and header tests, as well as in the string test when the variables extension is enabled, and in spamtest and virustest tests when those extensions are enabled
:regex	draft-murchison-sieve-regex-08	enable_sieve_regex MTA option	require "regex";	Regex match type
:value	RFC 5231		require "relational";	Match type for comparison on string values
test-name string	Private	FILTER_*test-name* mapping table		Custom test *test-name*: probes mapping table FILTER_*test-name* with *string*, evaluating to TRUE if mapping table sets $Y flag, or FALSE otherwise
Functions				
sub *name*[(*variables-list*)] { ... }				
	Private	system-level		(New in MS 8.0) Define a function/subroutine
my *var-name* [:= *value*]	Private			(New in MS 8.0) Declare a variable is local
return *value*	Private			(New in MS 8.0) Exit subroutine, returning *value*
translate *string-or-list source-charset dest-charset*				
	Private			(New in MS 8.0.1) Perform character set conversion, returning translated string or list
test-name string	Private	FILTER_*test-name* mapping table	require "variables";	Custom test *test-name* used as a function: probes mapping table FILTER_*test-name* with *string*, returning mapping's string result in variable ${0} and mapping's flag result in variable ${1}
...many-other-functions...	Private			See Symbol table functions
Values				
non-negative-integer	RFC 5228			Non-negative integer, range 0 to 2,147,483,647 (2^31 - 1); suffixes "K" (1024 or 2^10), "M" (2^20), or "G" (2^30) are permitted
negative-integer	Private			Negative integers are also supported by the MTA
"*utf-8-string*"	RFC 5228	max_sieve_string_size MTA option		String of characters represented in UTF-8; note that non-printing characters such as CR and LF are permitted, and so in particular strings can span multiple lines
"*character*"	RFC 5228			\\ represents the backslash character, \" represents the double quote character, and otherwise the backslash is ignored as if it were not present
text:*multiline text ended by CRLF.CRLF*	RFC 5228			More convenient way to enter multiline text strings; note must dot-stuff (as with SMTP messages)
["*s1*", "*s2*", ..., "*sN*"]	RFC 5228	max_sieve_list_size MTA option		List of strings
"${hex:*hex-pair-sequence*}"	RFC 5228		require "encoded-character";	*hex-pair-sequence* is one or more hexadecimal-encoded characters, space separated; *e.g.*, "{hex: 40 24}" represents "@$"
"${unicode: *unicode-hex-sequence*}"	RFC 5228		require "encoded-character";	*unicode-hex-sequence* is one or more Unicode hex-encoded characters, space separated; *e.g.*, "${unicode: 0000040}" represents "@"
${*variable-name*}	RFC 5229	max_variables MTA option	require "variables";	*variable-name* may be a variable identifier, or digit(s) identifying the variable, *e.g.*, ${a} refers to the variable named "a", while ${0} 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 *not* supported in "body" test arguments
expression	Private			The MTA supports a variety of string and numeric operators and functions, and supports expressions to compute values; see Operators in Order of Precedence and Symbol table functions
ADDRESS-PARTS				
:all	RFC 5228			Entire address
:comment	Private			Extract any parenthetical comment appearing after an address
:detail	RFC 3598		require "subaddress";	Extract subaddress (*e.g.*, folder name) in address and envelope tests
:display	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)
:domain	RFC 5228			Domain portion of address: the portion to the right of the @ character

:localpart	RFC 5228			RFC 822 "local-part" of address: the portion to the left of the @ character
:user	RFC 3598		require "subaddress";	Extract username (*i.e.*, local-part minus subaddress) in address and envelope tests
BODY-TRANSFORMS				
:raw	RFC 5173		require "body";	Interpret message body as an unprocessed "blob"; in particular, do not interpret MIME structure nor decode any content-transfer-encoding
:text	RFC 5173		require "body";	Decode MIME encoded text
COMPARATORS				
:comparator "i;ascii-casemap"	RFC 5228, RFC 4790			Treat uppercase and lowercase characters in the ASCII subset of UTF8 the same
:comparator "i;ascii-casemap-compress"	Private		require "comparator-i;ascii-casemap-compress";	(New in MS 8.0.1.2) Similar to "i;ascii-casemap", except that all internal folding white space characters (space, tab, carriage return, line feed) are replaced by a single ASCII space, and those at the beginining and end of the strings are removed. This is done to both the target and pattern strings prior to comparison
:comparator "i;ascii-casemap-collapse"	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
:comparator "i;ascii-integer"	Private		require "comparator-i;ascii-integer";	(New in MS 8.0) Test signed or unsigned integer values; (not supported on "body" tests)
:comparator "i:ascii-numeric"	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)
:comparator "i;octet"	RFC 5228, RFC 4790			Compare octets
:comparator "i;octet-compress"	Private		require "comparator-i;octet-compress";	(New in MS 8.0.1.2) Similar to "i;octet", except that all internal folding white space characters (space, tab, carriage return, line feed) are replaced by a single ASCII space, and those at the beginining and end of the strings are removed. This is done to both the target and pattern strings prior to comparison
:comparator "i;octet-collapse"	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
:comparator "i;utf-8"	?		require "comparator-i;utf8";	(New in MS 8.0) (not supported on "body" tests)
DATE-PARTS				
date	RFC 5260		require "date";	Date in "*yyyy*-*mm*-*dd*" format
day	RFC 5260		require "date";	Two digit day, "01", ..., "31"
hour	RFC 5260		require "date";	Two digit hour, "00", ..., "23"
iso8601	RFC 5260		require "date";	Date and time in restricted ISO 8601 format
julian	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
milliunixepoch	Private		require "date";	(New in 8.1.0.6) Date and time in integer milliseconds since Unix epoch
minute	RFC 5260		require "date";	Two digit minute, "00", ..., "59"
month	RFC 5260		require "date";	Two digit month, "00", ..., "12"
second	RFC 5260		require "date";	Two digit second, "00", ..., "60"
std11	RFC 5260		require "date";	Date and time in a format appropriate for use in a Date: header field (RFC 822 format)
time	RFC 5260		require "date";	Time in "*hh*:*mm*:*ss*" format
unixepoch	Private		require "date";	(New in 8.1.0.6) Date and time in integer seconds since Unix epoch

weekday	RFC 5260		require "date";	Day of the week expressed in range "0" (Sunday) to "6" (Saturday)
year	RFC 5260		require "date";	Four digit year, "0000", ..., "9999"
zone	RFC 5260		require "date";	Time zone in use, in +*hh*:*mm* or -*hh*:*mm* format
ENVELOPE-PARTS				
from	RFC 5228		require "envelope";	Envelope From address
to	RFC 5228		require "envelope";	Envelope To address
auth	Private		require ["envelope", "envelope-auth"];	Access message's AUTH value
conversiontag	Private	system-level	require "envelope";	In system-level Sieve, access message's conversion tag(s)
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 ldap_default_domain MTA option, next the value of received_domain MTA option, or if neither are defined the value of the L channel official_host_name
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.max_mime_width	Private		require "environment";	(New in MS 8.0.2.2) The maximum number of MIME parts found in any multipart in the message.
vnd.oracle.message-hash	Private		require "environment";	(New in MS 8.0) Return any message hash calculated by the generatemessagehash channel option; the query will fail if no message hash was generated
vnd.oracle.mime_levels	Private		require "environment";	(New in MS 8.0.2.2) The number of accessible MIME levels in the current message, i.e. the MIME "depth" of the message.
vnd.oracle.mt-priority	Private		require "environment";	(New in MS 8.0) Match against message's current MT-PRIORITY value
vnd.oracle.notifycount	Private		require "environment";	(New in MS 8.0.1.2) Return the count of the number of notification messages the current sieve has enqueued.
vnd.oracle.notifyquota	Private		require "environment";	(New in MS 8.0.1.2) Return the number of additional notify actions the current sieve is able to perform.
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.oracle.vacationcount	Private		require "environment";	(New in MS 8.0.1.2) Return the count of the number of vacation messages enqueued by the current sieve. Note that this is not necessarily the same as number of vacation actions performed since any number of conditions can turn a vacation action into a no-op.
vnd.oracle.vacationquota	Private		require "environment";	(New in MS 8.0.1.2) Return the number of additional vacation actions the current sieve is able to perform.
vnd.oracle.warnings	Private		require "environment";	(New in MS 8.1.0.3) Returns the text of any warnings that have been issued since any previous environment test of this item, or since the start of script execution if no previous tests have been made. The environment call fails if no warnings are available to return.
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
custom-item-name	Private	$+E*item-name*|*item-value* in an *_ACCESS address mapping table	require "environment";	Arbitrary, custom environment items for a message may be set from the FROM_ACCESS or *_ACCESS recipient accress mapping tables, for subsequent use in environment tests
MATCH-TYPES				
:aindex	Private			Match against Nth address on header line
:contains	RFC 5228			Substring match
:count	RFC 5231		require "relational";	Match type for numeric comparison on number of occurrences; (not available for use with "body" test)
:index	RFC 5260		require "index";	Match against Nth named header line for address, header, or date tests
:last	RFC 5260		require "index";	For the :index modifier, count backwards from the end of the message header
:is	RFC 5228			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, importancetest, memcache, metermaid, spamtest, string, and virustest, and on deleteheader and replaceheader actions
:matches	RFC 5228			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 are 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. As of MS 8.1.0.1, the "`deleteheader`" action has been extended to allow glob-style wildcards in the field name. (Obviously this capability should be used with great care, as not only does it make it possible to delete entire swathes of header information, it also is an expensive operation to perform. In particular, leading wildcards in the field name should be avoided if at all possible.) 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)
- Fcc extension (draft-ietf-extra-sieve-fcc-02.txt). This extension adds support for delivering a copy of any generated message to the `notify` and `vacation` actions to the mailbox specified by a :fcc nonpositional parameter.
- 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 MS 8.0 is support for the "`extracttext`" and "`foreverypart`" Sieve extensions from RFC 5703. New in MS 8.1 is the ability to control whether or not the foreverypart Sieve control looks inside of nested messages or treats them as leaf parts. The `:processnestedmesssages` argument tells foreverypart to look inside and is the default. `:retainnestedmessages` causes nested messages to be treated as leaf parts. (8.0, 8.1)
- Notify extension (RFC 5435) and the mailto notification method (RFC 5436). Notify with the mailto method provides the means to send an 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). As of MS 8.0.1.3, :count may also be used as a modifier to other match types in header and address tests to count and test the number of matches that occurred.
- 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 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. (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 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 "spamtest" and "spamadjust" extensions allow multiple Sieve scripts to cooperate in determing 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 "spamtest" and "virustest" test, respectively, would then test. (6.0)
- The "strongrandom" function takes as an 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 of performance concerns).
- The "random" function takes as an argument an integer specifying an upper limit n on the range of values to return; a uniformly distributed random number between 0 and n-1 is then returned. The linear congruential generator described in "Random Number Generators: Good Ones Are Hard To Find", S. Park and K. Miller, CACM 31 No. 10, pp. 1192-1201, October 1988 is used to generate these numbers; a separate sequence seeded from the system time is used for each sieve evaluation. An explicit 32 bit integer seed can be specified by calling the "randomssed" function. (This may be useful for debugging purposes.) A value of 0 will cause the sequence to be reinitialized from the system clock.
- 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 MS 6.2 and updates to MS 6.3)
- Subroutines are supported. (8.0)
- Trailing commas are now allowed in string lists. (8.1.0.1)

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 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.
- 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 effects similar to "`discard`" or "`jettison`" 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 `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.

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 warnng 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 MS 6.1, and the standard deleteheader action and proposed replaceheader action since 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, 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 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`".

New in 8.0.1.2, the MTA supports new Sieve environment items, `vnd.oracle.notifycount`, `vnd.oracle.notifyquota`, `vnd.oracle.vacationcount`, `vnd.oracle.vacationquota` The "count" items return the number of notification or vacation messages the current sieve has enqueued. The "quota" items return the number of additional notification or vacation actions the current sieve is allowed to perform.

Two additional private Sieve environment items have been added in MS 8.0.2.2. `vnd.oracle.mime_levels` returns the number of accessible MIME levels in the current message, i.e. the MIME "depth" of the message. `vnd.oracle.max_mime_widh` returns the maximum number of MIME parts found in any multipart in the message.

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|*       $C$+Eheloname|$2
```

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, or when the acceptalladdresses channel option is set), 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 (MS 6.1), "refuse" was only supported in system-level Sieve scripts. That restriction was removed for 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.*, sytem-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 `$:S` (flag set) or `$;S` (flag clear) flag test syntax.

The mapping template must set the `$Y` flag if the test succeeds and $N if the test fails. $F may be set to report an error; in this case the mapping result will be used as the error message.

Failure to set $Y, $N, or $F will cause an "unknown list" error to be signaled.

Note that $Y takes precedence over $N, so a single mapping template can set $N before performing a lookup, $Y after, and both the success and failure case will be handled.

In the case of a successful `redirect` action, the template should return a URL pointing to the desired list of addresses.

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 multiple recipient 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`" with a test 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`, `envelope`, or `header` 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_4` setting of

```
# msconfig set ldap_spare_4 psroot
```

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

```
LDAP_SPARE_4=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|*   \
$N$]pab$1?piEmail1?sub?(|(piEmail1=$=$4$_)(piEmail2=$=$4$_)(piEmail3=$=$4$_))[$Y
  envelope|*|*|*|*|pab|*  \
$N$]pab$1?piEmail1?sub?(|(piEmail1=$=$4$_)(piEmail2=$=$4$_)(piEmail3=$=$4$_))[$Y
  header|*|*|*|*|pab|*    \
$N$]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_4|address-from-message|piEntryID
!
  GROUP|address|*|*|%*    \
$N$]pab$0?piEmail1?sub?(&(memberOfPiGroup=$2$3)(|(piEmail1=$=$1$_)(piEmail2=$=$1$_)(piEmail3=$=$1$_)))[$Y
  GROUP|envelope|*|*|%*    \
$N$]pab$0?piEmail1?sub?(&(memberOfPiGroup=$2$3)(|(piEmail1=$=$1$_)(piEmail2=$=$1$_)(piEmail3=$=$1$_)))[$Y
  GROUP|header|*|*|%*    \
$N$]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 `spamfilterN_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|*   $N$C${HOLIDAY|$4}$Y$E
  date|*|*|*|*|holiday|*          $N$C${HOLIDAY|$4}$Y$E
```

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.

Sieve external list tests don't have to involve checking an actual list. In such cases a string test is typically used since no value actually comes from the message. For example, the mechanism can be used to implement a user attribute which, if present, is inserted into all messages delivered to the user as a header. The first step is to make the attribute, which we'll call *userAddHeader*, available to the external list machinery as a spare attribute:

```
# msconfig set ldap_spare_4 userAddHeader
```

The SIEVE_EXTLISTS mapping would then be:

```
SIEVE_EXTLISTS

    string|*||*|*|addheader|*    $N$*
    string|*|*|*|*|addheader|*   $Y$1$*
```

Note the use of $* to make this sieve recipient-specific.

And the sieve to perform the header addition would be:

```
require ["extlists", "editheader", "variables"];
if string :list "addheader" "addheader" {
  addheader "X-Added-Header" "${1}";}
```

This would probably best be implemented as a destination channel sieve, although any type of sieve would work.

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 editting 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
  envelope|*|*|*|*|capture-to|*    $N
!
! When the probe matches, the test succeeds ($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|*@*  $N$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:///$1?mailCaptureInternet?sub?(&(|(mail=$=$0$_)(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 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 capabability 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.

New in MS 8.1 is the ability to control whether or not the foreverypart Sieve control looks inside of nested messages or treats them as leaf parts. The `:processnestedmesssages` argument tells foreverypart to look inside and is the default. `:retainnestedmessages` causes nested messages to be treated as leaf 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 MS 7.0.5, the MTA supports RFC 5435 (Sieve Email Filtering: Extension for Notifications) and RFC 5436 (Sieve Notification Mechanism: mailto). As of MS 8.0, a private `nonotify` system-level action is also provided. And finally, as of MS 8.0.2.3, the File Carbon Copy (Fcc) described in the Internet-Draft draft-ietf-extra-sieve-fcc-02.txt is also supported.

5.1.2.16.1 The notify extension

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 MS 6.2) for a subset of the "`notify`" action defined in draft-martin-sieve-notify-01.txt.

The limitations on the implementation of draft-martin-sieve-notify-01.txt are:

1. The message parameter is not optional.
2. The "$...$" substitutions defined in section 3.1 are not supported. (Sieve variables may be used to provide this functionality.)
3. The "denotify" action is not supported.

Both forms of "`notify`" can be used simultaneously; *e.g.*:

```
require ["enotify","notify"];
# New, standard form -- the enotify 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.

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.

5.1.2.16.2 The notify_method_capability and valid_notify_method tests

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).

5.1.2.16.3 The nonotify action

New in 8.0, the MTA supports a private action 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.)

5.1.2.16.4 The :mime nonpositional parameter

Both forms of the notify support support the nonpositional :mime parameter. If specified this parameter has the same semantics as with the vacation action: The causes the specified message body to be treated as a MIME entity as defined in RFC 2045 section 2.4, including both MIME headers and content.

5.1.2.16.5 The fcc extension

The `fcc` extension defines a new optional tagged argument ""fcc" that can be used with the notify and vacation actions to allow a copy of the vacation or notification message to be filed into a target mailbox belonging to the sieve owner. The mailbox argument to :fcc is analogous has the same semantics as the mailbox argument to the fileinto action; in effect it is as if a copy of the message was sent to the sieve owner with a user sieve specifying a fileinto action.

Support for the `fcc` extension was added in MS 8.0.2.3.

5.1.2.17 Sieve override extension

Multiple sieves can be and often are evaluated for a given message recipient. When this happens some actions, such as `addheader`, `capture`, or `addconversiontag`, are accumulated across all applicable sieves. But actions that determine the ultimate disposition of a message have to come from a single sieve.

The general rule is that the most specific sieve that sets a disposition wins. So if, say, a keep done by a system sieve would be overridden by a discard done by a user sieve.

A couple of nonstandard actions have previously been added to override this determination order. In particular, the most general sieve that performs a `jettison` or `refuse` action will determine the disposition of a message for a given recipient.

The use cases for `jettison` and `refuse` are obvious: If system policy has determined that a user cannot see a message, the user's own preferences need to be overridden. The jettison and refuse actions also have the advantage that they are incompatible with other disposition actions, making them easy to use.

In contrast, the other disposition actions (keep, discard, fileinto, redirect, etc.) are all compatible with each other, and in practice are used in a wide variety of combinations and permutations. So separate "override" variants of these actions don't make as much sense.

Even so, it may be useful for a system-level sieve to forcibly direct messages using redirect and possibly fileinto, while retaining their standard semantics in the context of the sieve where they are specified. For this to work correctly, such sieves need to be able to override disposition actions specified by more specific sieves. Accordingly, a nonstandard sieve override extension and corrresponding action has been added. A sieve that uses this action becomes the sieve determining the disposition of the message. If multiple sieves employ override, the most general one wins.

For example, suppose you have a system sieve:

```
redirect "foo@bar";
```

And the recipient has a user sieve:

```
keep;
```

The `keep` action wins and the message is filed into the recipient's INBOX folder. Now suppose the system sieve is changed:

```
require "override"; override; redirect "foo@bar":
```

Now the redirect action wins and the message is redirected.

5.1.2.18 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 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 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 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`".

5.1.2.19 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"
```

As of MS 8.0.1.3, the MTA allows :count to be combined with other match-types in header and address tests. When this is done the test performs the non-count match first, counting the number of matches. The resulting count is then compared with a third argument. An "i;ascii-numeric" comparator is always used for this second match.

For example, the following test checks to see if the domain "example.com" appears in more than five Received: fields and holds the message if it does:

```
if header :count "gt" :contains "received" "example.com" "5" {hold;}
```

Note that the address test counts addresses, not fields. For example, the following test checks to see if there are less than 10 addresses that have a subdomain of example.org as their domain in a recipient field:

```
if address :count "lt" :matches
```

```
:domain ["to", "cc", "bcc"] "*.example.org" "10" {...}
```

5.1.2.20 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 using the "`spamadjust`" action via a corresponding pair of MTA options `spamfilterN_verdict_M` and `spamfilterN_action_M`:

```
msconfig> exec get_path "config"
> "/opt/SUNWmsgsr/config"
msconfig> show spamfilter1_*
role.mta.spamfilter1_config_file = IMTA_TABLE: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
```

5.1.2.21 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`".

5.1.2.22 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 (a 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.

As of MS 8.0.2.3, the MTA supports a private argument ":noheadercheck" that suppresses the MTA's normal requirement (per the RFC 5230, Section 4.6 requirement) that various header fields, e.g., List-Id:, not be present for a vacation reply to the 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 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.

Support for the fcc extension was added in MS 8.0.2.3. The fcc extension defines a new optional tagged argument ":fcc" that can be used with the notify and vacation actions to allow a copy of the vacation or notification message to be filed into a target mailbox belonging to the sieve owner. The mailbox argument to :fcc is analogous has the same semantics as the mailbox argument to the fileinto action; in effect it is as if a copy of the message was sent to the sieve owner with a user sieve specifying a fileinto action.

5.1.2.22.1 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.

5.1.2.23 Sieve variables extension

As of MS 6.2, the MTA added initial support for the Sieve variables extension, modified in 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.
-
-

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.

5.1.2.24 Sieve conversiontag extensions

For system-level Sieve scripts, the MTA's private conversion tag manipulation Sieve actions "`addconversiontag`" and "`setconversiontag`" have been supported since 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 converison tag setting mechanisms. "`removeconversiontag`" can be used to undo all or part of preceeding "`addconversiontag`" or "`setconversiontag`" operations.

Also, in system-level Sieves, as of 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.

5.1.2.24.1 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 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 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"; }
```

5.1.2.25 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 discussion of the Sieve editheader extension.

5.1.2.26 Sieve addtag extension

The MTA's private Sieve extension `addtag` action has been supported since 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 MS 6.1p1 and 6.2).

Adding multiple tags is supported, for instance, "`Re:`" and "`FWD:`". Prior to MS 6.3, `addtag` took a space-delimited list of arguments; as of 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 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 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).

5.1.2.27 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 milter, 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.

5.1.2.28 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 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`".

5.1.2.29 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.

As of MS 8.0.2.2, the "hold" action accepts an optional string argument. This argument is used to specify a reason string for the hold action; the value will appear in the log_filter field of MTA message transaction log entries.

5.1.2.30 Sieve comparators

In addition to the standard Sieve comparators "i;ascii-casemap" and "i;octet" described in RFC 5228, 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.

New in MS 8.0. 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 optional semantically meaningless 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.

New in MS 8.0.1.2. The MTA's comparators "i;ascii-casemap-compress" and "i;octet-compress" are similar to the correspondingly named standard comparators, except that all folding white space characters (space, tab, carriage return, line feed) are compressed into a single ASCII space and those at the begininng and end of the strings are removed. This is done to both the target and pattern strings prior to comparison. Use of these white-space-compressing comparators is recommended for Sieve comparisons of structured header values that include embedded white space.

5.1.2.31 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 "spamtest" 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 "spamtest" 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.

5.1.2.32 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);
      ...
      nextif (expression);
      ...
      }
```

A loop may contain zero or more "exitif" and/or "nextif statements. The loop terminates if the argument to exitif evaluates to true. New in 8.0.2.3, nextif can be used to cause the loop to restart if the associated condition evaluates to true.

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.

5.1.2.33 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:

`:host host-string`	Specifies the host name(s) and port(s) for the memcached server to use in *`host`*`:`*`port`* format. The *`port`* part is optional and defaults to value given to the `memcache_port` MTA option. If "`:host`" isn't specified, the host and port are given by the `memcache_host` and `memcache_port` MTA options respectively.
`:host [host-string1, host-string2, ...]`	If a 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 mapping callout.
`:tableprefix prefix-string`	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. "`:tableprefix`" 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 "`:tableprefix`" parameter may be useful in specifying such structure.)
`:timeout timeout-number`	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.
`:quota quota-int`	Specifies the maximum number of counts to permit during the quota timeout period. This corresponds to MeterMaid throttle table `quota` setting.
`:quotatimeout quota-timeout-int`	Specifies the duration in seconds of the period over which counts are recorded. This corresponds to MeterMaid throttle table `quota_time` setting.
`:penalize`	Corresponds to MeterMaid throttle table option `penalize` setting.
`:limit`	(New in MS 8.0.2.3.) Normally :throttle operations update the throttle even when the throttle engages; this is appropriate when throttling an external activity that occurs outside the control of Messaging Server. The addition of :limit causes a :throttle test to only update when the throttle does not engage; this is appropriate when a throttle is being used to limit some internal activity that doesn't occur when the throttle engages. Any operation failure is also handled in a failsafe manner when :limit is specified

	- server down, network failure, wrong data type, etc. - to return TRUE rather than the usual FALSE.
`: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 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 suceeds 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 `"i;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 :release [:duplicate]
         [:host host-string] [:tableprefix prefix-string]
         [:timeout timeout-value]  key-string
```

Implements a generic reservation capability, where a specified key-string can be repeatedly reserved up to a specified quota. :release releases a reservation previously created by :reserve. The associated memcache entry contains a record of how many reservations are held by each process on the host. Outstanding reservations associated with nonexistent processes will be automatically deteled.

This operation always functions as a test, returning TRUE if the reservatation has been releaves succesfully, FALSE otherwise.

memcache reservation operations are performed on 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 :reserve :quota quota-numeric [:duplicate]
         [:host host-string] [:tableprefix prefix-string]
         [:timeout timeout-value]  key-string
```

Implements a generic reservation capability, where a specified key-string can be repeatedly reserved up to a specified quota. Each reserve operation must be matched by a corresponding release. The associated memcache entry contains a record of how many reservations are held by each process on the host. Outstanding reservations associated with nonexistent processes will be automatically deteled.

Note that since there is no server-side awareness of entry semantics the quota parameter must be specified in every reserve call in case the entry needs to be created. If the entry already exists the quota parameter value will be checked against the corresponding value stored in the entry.

Each reserve operation increments the reservation count by 1, and should be matched by a corresponding release operation.

This operation always functions as a test, returning TRUE if the reservation is successful, FALSE otherwise.

memcache reservation operations are performed 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] [:limit]
         [: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.)

New in MS 8.0.2.3, "`:limit`" can be specified to prevent the throttle from being adjusted if it engages and to behave in a failsafe fashion.

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 adjusted 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.

5.1.2.34 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.

5.1.2.35 Sieve redis extension

As of the 8.0.2.3 release, the MTA supports a private "redis" operator, which may be used to perform both actions and tests. (Configuration of MTA connections to Redis is required.) This nonstandard extension is available without any special capability declaration; no "require" action is needed for its use. By default, the "redis" operator may be used in any Sieve script, but its availability may be selectively disabled via the enable_sieve_redis MTA option. This extension is provided so that Sieve filters can access and manipulate data using the redis protocol.

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

:tableprefix prefix-string	Unlike MeterMaid, each redis server provides a single storage area; in effect a single "table". Multiple tables can be implemented by adding a prefix to the key. ":tableprefix" 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.
:timeout timeout-number	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.
:quota quota-int	Specifies the maximum number of counts to permit during the quota timeout period. This corresponds to MeterMaid throttle table quota setting.

`:quotatimeout quota-timeout-int`	Specifies the duration in seconds of the period over which counts are recorded. This corresponds to MeterMaid throttle table `quota_time` setting.
`:penalize`	Corresponds to MeterMaid throttle table option `penalize` setting.
`:limit`	Normally :throttle operations update the throttle even when the throttle engages; this is appropriate when throttling an external activity that occurs outside the control of Messaging Server. The addition of :limit causes a :throttle test to only update when the throttle does not engage; this is appropriate when a throttle is being used to limit some internal activity that doesn't occur when the throttle engages. Any operation failure is also handled in a failsafe manner when :limit is specified - server down, network failure, wrong data type, etc. - to return TRUE rather than the usual FALSE.
`:set`	New in MS 8.1.0.3. Specifies that the data associated with the key is a Redis set.
`:sortedset`	New in MS 8.1.0.3. Specifies that the data associated with the key is a Redis sorted set.
`:score score-number`	New in MS 8.1.0.3. Specifies the score for an entry in a Redis sorted set. The default score is 0.
`: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 redis Sieve extension provides the following operations:

```
redis :add [:tableprefix prefix-string] [:duplicate]
      [:timeout timeout-number] [:set | :sortedset]
      [:score score] key-string value-string
```

Add a new entry with the specified key, value and timeout. The operation only suceeds if the entry does not already exist. The operation returns TRUE if the entry is added successfully, FALSE if not. If `:set` or `:sortedset` are specified add the specified value to the set. The set will be created if it doesn't already exist. A score of 0 is used for entries assed to sorted sets if no other value is specified.

For example:

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

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

```
redis :adjustdown adjustment-value [:tableprefix prefix-string]
      [:duplicate] [MATCH-TYPE] [COMPARATOR]
      [:quotatimeout quotatimeout-numeric [:penalize]]
      [:timeout timeout-value] [:sortedset] 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.

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.

Redis "`: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.

New in MS 8.1.0.3, Redis "`:adjustdown`" operations can be applied to members of sorted sets by specifying the `:sortedset` parameter. If this is done `test-string` is interpreted as the name of the set member.

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

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

```
redis :adjustup adjustment-value [:tableprefix prefix-string]
      [:duplicate] [MATCH-TYPE] [COMPARATOR]
      [:quotatimeout quotatimeout-numeric [:penalize]]
      [:timeout timeout-value] [:sortedset] 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.

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.

Redis "`: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.

New in MS 8.1.0.3, Redis "`:adjustup`" operations can be applied to members of sorted sets by specifying the `:sortedset` parameter. If this is done `test-string` is interpreted as the name of the set member.

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

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

```
redis :append [: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.

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

```
redis :expire [:tableprefix prefix-string]
      [:duplicate] key-string
```

Sets the expiration time in seconds for the entry with the specified key. A time of 0 will cause the entry to be be deleted immediately. Negative values will cause the entry to be preserved indefinitely.Remove the entry with the specified key. Returns TRUE if the operation is successful, FALSE if it is not. Note that the expiration time for sets and sorted sets can be set without specifying any additional parameters.

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

```
redis :fetch [:tableprefix prefix-string]
      [MATCH-TYPE] [COMPARATOR] [:set | :sortedset]
      [:withscores] 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 `"i;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 redis :fetch :matches "entry" "*" { ... entry exists ... }
```

The members of Redis sets and sorted sets may be fetched by specifying the corresponding parameter. Multiple members are returned as lists. If `:withscores` is specified member-score pairs are returned.

redis "`:fetch`" operations are simply repeated on reevaluations.

```
redis :release [:duplicate] [:tableprefix prefix-string]
      [:timeout timeout-value]  key-string
```

Implements a generic reservation capability, where a specified key-string can be repeatedly reserved up to a specified quota. :release releases a reservation previously created by :reserve. The associated Redis entry contains a record of how many reservations are held by each process on the host. Outstanding reservations associated with nonexistent processes will be automatically deteled.

This operation always functions as a test, returning TRUE if the reservatation has been releaves succesfully, FALSE otherwise.

Redis reservation operations are performed on reevaluations.

```
redis :remove [:tableprefix prefix-string]
      [:duplicate] key-string
```

Remove the entry with the specified key. Returns TRUE if the operation is successful, FALSE if it is not. Note that sets and sorted sets can be removed without specifying any additional parameters.

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

```
redis :replace [: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.

redis "`:replace`" operations are simply repeated on reevaluations.

```
redis :reserve :quota quota-numeric [:duplicate]
      [:tableprefix prefix-string]
      [:timeout timeout-value]  key-string
```

Implements a generic reservation capability, where a specified key-string can be repeatedly reserved up to a specified quota. Each reserve operation must be matched by a corresponding release. The associated redis entry contains a record of how many reservations are held by each process on the host. Outstanding reservations associated with nonexistent processes will be automatically deteled.

Note that since there is no server-side awareness of entry semantics the quota parameter must be specified in every reserve call in case the entry needs to be created. If the entry already

exists the quota parameter value will be checked against the corresponding value stored in the entry.

Each reserve operation increments the reservation count by 1, and should be matched by a corresponding release operation.

This operation always functions as a test, returning TRUE if the reservation is successful, FALSE otherwise.

redis reservation operations are performed on reevaluations.

```
redis :store [:tableprefix prefix-string]
      [:timeout timeout-value] [:set | :sortedset]
      [:score score] 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.

Store operations are identical to add operations for sets and sorted sets.

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

```
redis :throttle :quota quota-numeric [:duplicate]
      :quotatimeout quotatimeout-numeric [:penalize] [:limit]
      [: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.)

New in MS 8.0.2.3, "`:limit`" can be specified to prevent the throttle from being adjusted if it engages and to behave in a failsafe fashion.

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 adjusted value. The default MATCH-TYPE is `:value "gt"` and the default COMPARATOR is `"i;ascii-numeric"`.

redis "`: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.

5.1.2.36 Sieve regex extension

As of 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.

5.1.2.37 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.

5.1.2.38 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 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.

5.1.2.39 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.

5.1.2.40 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.

5.1.2.41 Sieve transactionlog extension

New in 8.0, for system-level Sieve scripts, the MTA's private "`transactionlog`" Sieve action can be used to annotate the transaction log. The action takes a single string argument specifying the annotation text.

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 for details.

5.1.2.42 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");
```

5.1.2.43 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.

New in MS 8.1.0.3, the private Sieve environment item, vnd.oracle.warnings, allows scripts to check and see if any Sieve warnings have occurred. The operation only succeeds if one or more warnings have occurred since the last check of vnd.oracle.warnings - or if no previous calls have been made, since the start of script processing. The item return value is the text of those warnings.

5.1.2.44 Sieve custom tests via mappings

As of updates to MS 6.2 and updates to 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";}
  }
}
```

5.1.2.45 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 preceeding 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.

5.1.2.46 Sieve expressions

The MTA's Sieve implementation supports the use of expressions in places where the base Sieve specification expects values. Such expressions can make use of a number of arithmetic and string functions and operators.

Note that because Sieve syntax uses square brackets to denote lists, the MTA's usual support for using square brackets to index into strings is not supported within Sieve filters; within Sieve filters, parentheses must be used rather than square brackets to index into a string, *i.e.*, `a(1)` rather than `a[1]`.

Note that the `imsimta test -expression` utility can be used to test expression evaluation, and in particular Sieve filter expression evaluation.

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 *`CONFIGROOT/`*`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 `$S` 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 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:

- `discard`,
- `fileinto`,
- `keep`,
- `redirect`,
- `reject`, and
- (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:
   - `jettison`(like discard) and
   - `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.

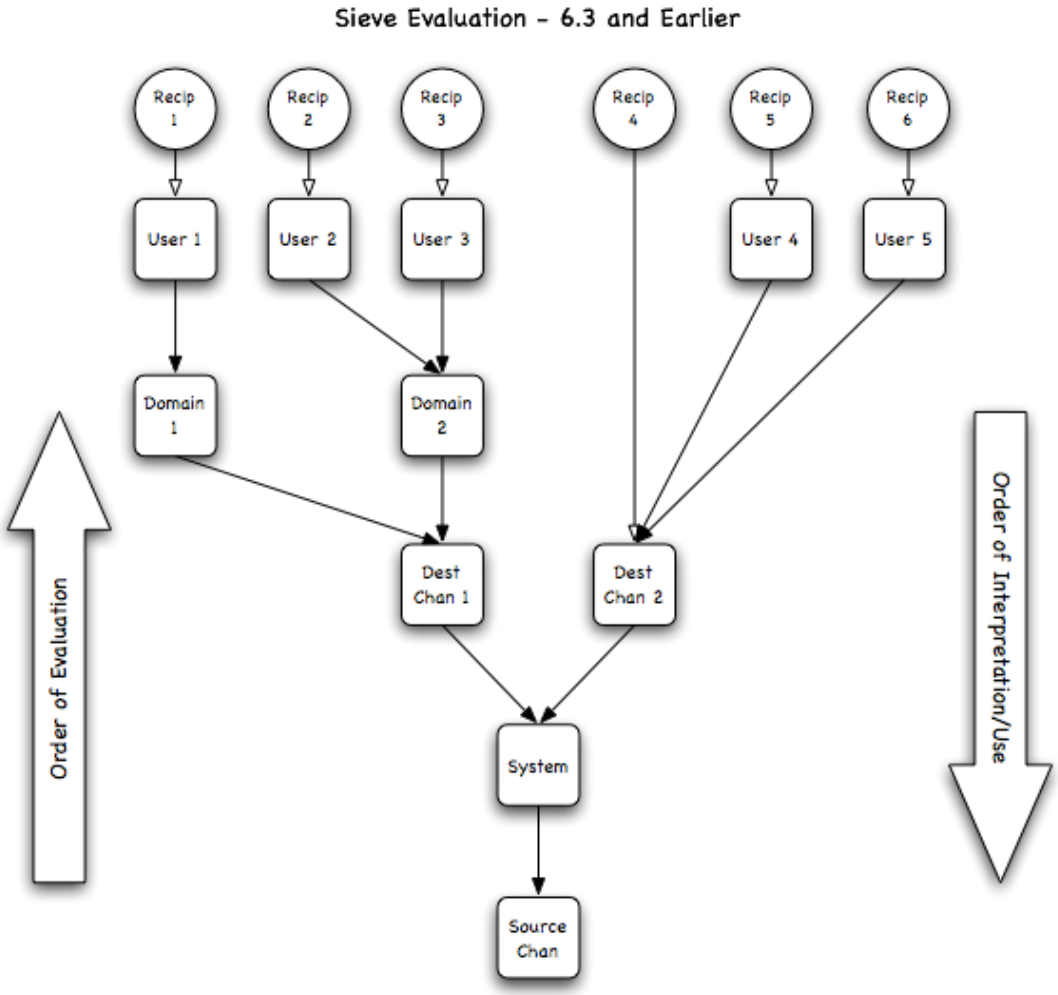
Prior to the 7.0 release, "`refuse`" actions were only available to system-level Sieve scripts, and "`refuse`", when used, forced the return of a 5yx response to the DATA command in SMTP. In 7.0 and later this is no longer the case: "`refuse`" now behaves like "`jettison`", making it possible for it 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).

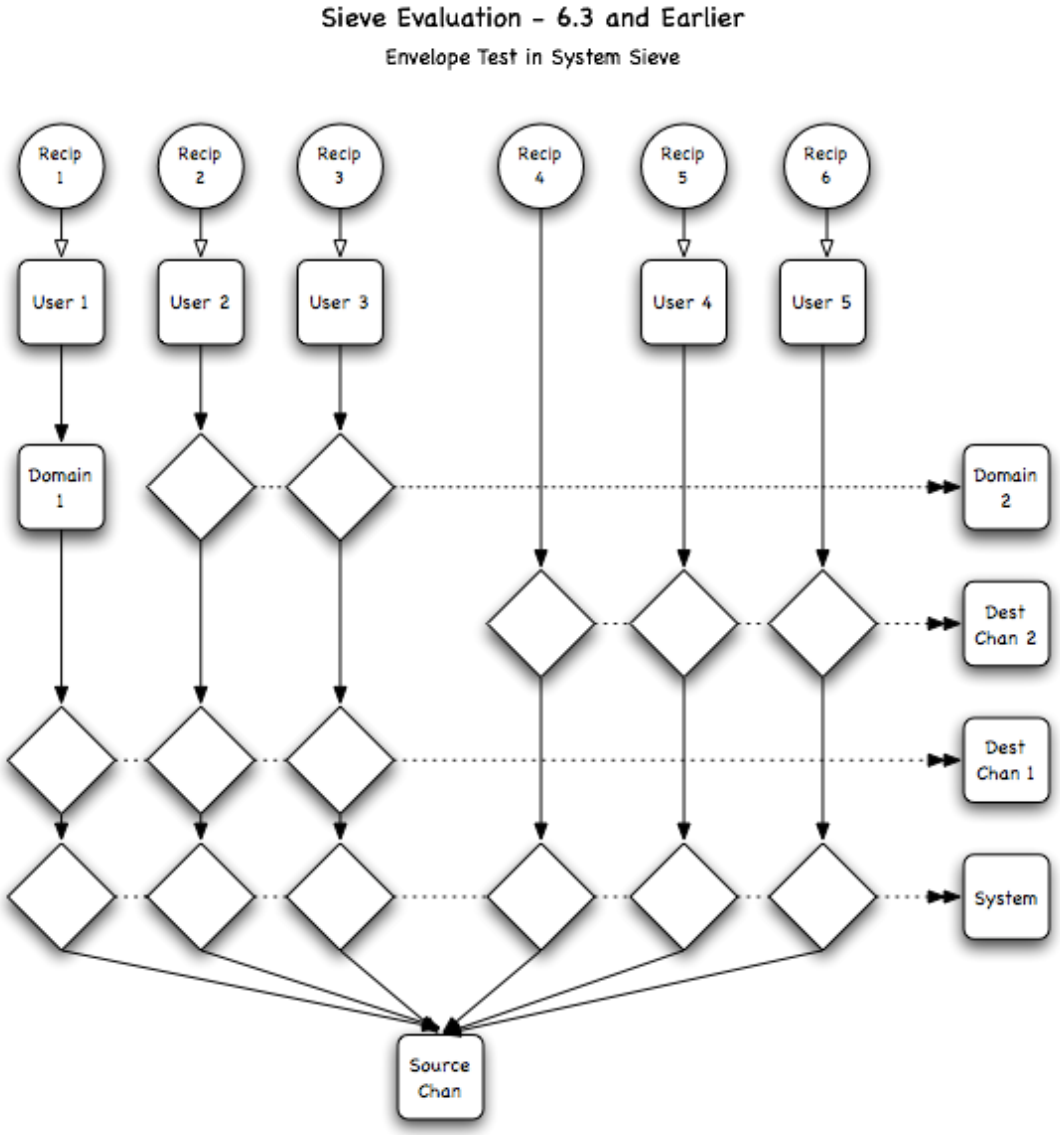
The prior-to-7.0 organization just described was fine in terms of Sieve script evaluation semantics. However, after it was implemented and various additional Sieve extensions were defined, a number of problems emerged:

- 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"`, `"spamtest"`, 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 called for the "`envelope`" extension, the recipient tree shown above ended up looking like this:

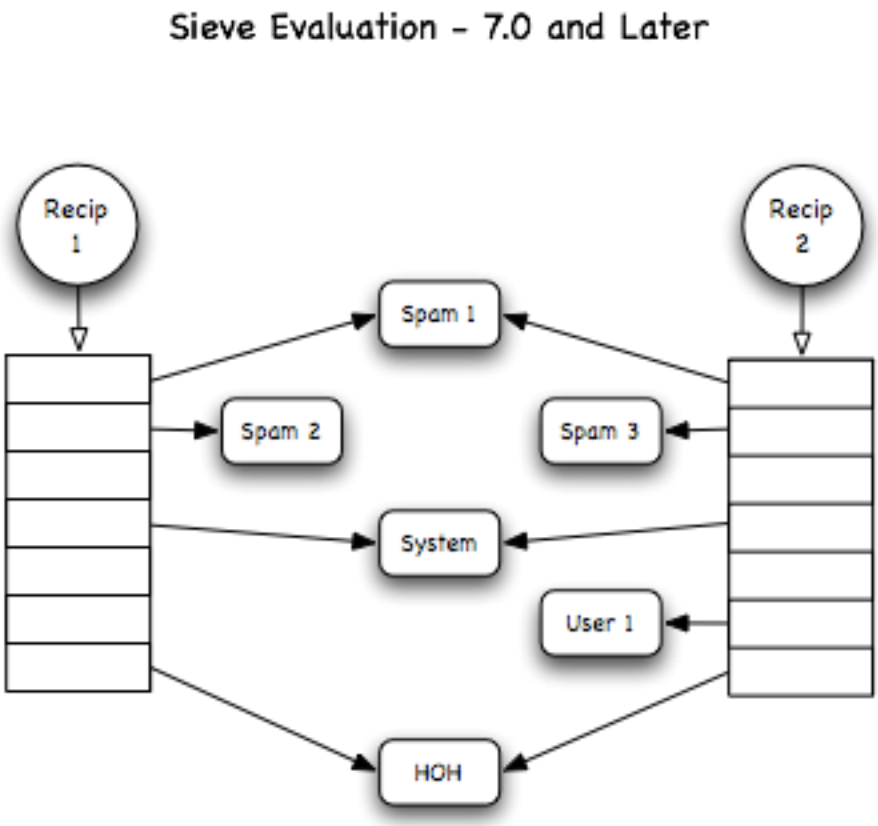


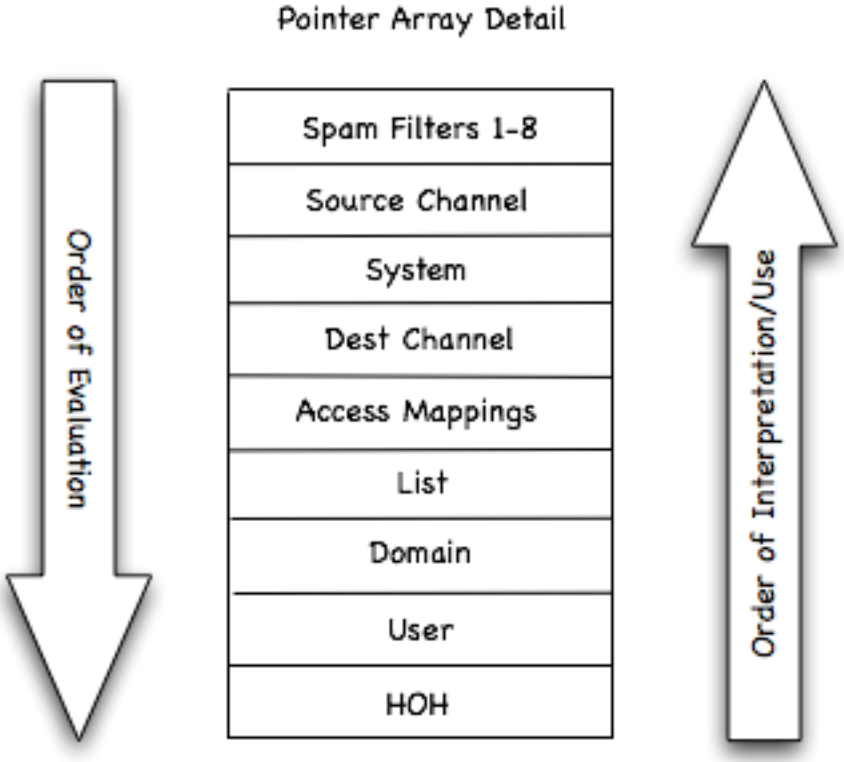
- All Sieve scripts must list all of the extensions they employ in an initial require clause. However, it is common for an extension to be listed but not actually used - and this is not necessarily a result of poor coding practice. For example, a script might perform a header or address test and depending 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.

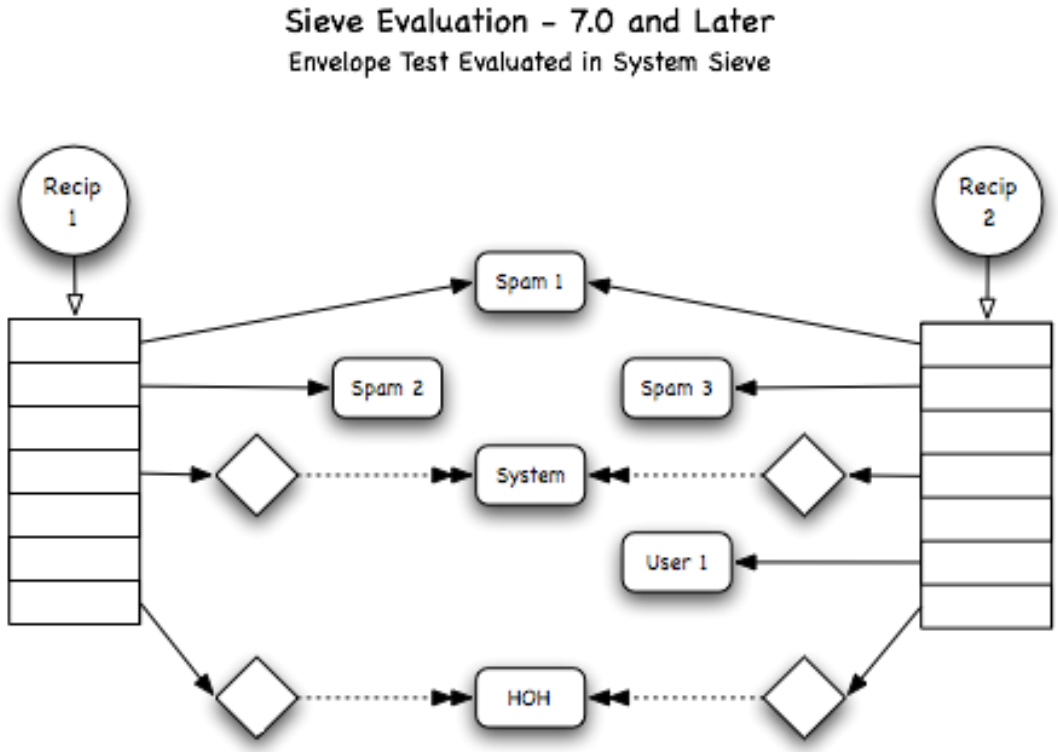
A new way of linking scripts to users was needed and has been implemented in 7.0. The linkage tree is gone, replaced with a per-recipient array:


Sieve Evaluation - 7.0 and Later
Recip 1
Recip 2
Spam 1
Spam 2
Spam 3
System
User 1
HOH


Pointer Array Detail
Order of Evaluation
Spam Filters 1-8
Source Channel
System
Dest Channel
Access Mappings
List
Domain
User
HOH
Order of Interpretation/Use

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 poceeds 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:



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

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 Tcpd 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-specification*s. 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-pattern*s, and TCP wrapper filter EXCEPT operator for further details on the EXCEPT operator.

client-list is a comma or space separated list of *client-specification*s. 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.

```
+imaps,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 *and* 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.
`DNSSPOOFER`	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`. Note that `255.255.255.255` is not permitted as a mask; use CIDR notation with `/32` instead.
- 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 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, 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 attribute[1] and the domain level `mailDomainAllowedServiceAccess` LDAP attribute.

[1] The 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.

6.4 Component domainallowed and domainnotallowed options

Various components allow setting allow or deny TCP access filters via *component*`.domainallowed` or *component*`.domainnotallowed` options. (However, the MMP and its subcomponents instead use a general `tcpaccess` option.)

6.4.1 `domainallowed` Option

The `domainallowed` option (available under `ens`, `eval_ldapd`, `http`, `imap`, `imapproxy`, `pop`, `popproxy`) specifies access filters specifying which domains and/or IP addresses are allowed access for the selected server.

6.4.1.1 Use with the IMAP proxy

The `domainallowed` IMAP Proxy/POP Proxy option (also available under `ens`, `eval_ldapd`, `http`, `imap`, and `pop`) specifies access filters specifying which domains and/or IP addresses are allowed access for the selected server.

6.4.1.2 Use with the POP proxy

The `domainallowed` IMAP Proxy/POP Proxy option (also available under `ens`, `eval_ldapd`, `http`, `imap`, and `pop`) specifies access filters specifying which domains and/or IP addresses are allowed access for the selected server.

6.4.1.3 Use with http

The `domainallowed` MSHTTP option specifies access filters specifying which domains and/or IP addresses are allowed HTTP access.

6.4.1.4 Use with imap

The `domainallowed` IMAP option specifies access filters specifying which domains and/or IP addresses are allowed IMAP access.

6.4.1.5 Use with pop

The domainallowed POP option specifies access filters specifying which domains and/or IP addresses are allowed POP access.

6.4.1.6 Use with ens

The domainallowed ENS option specifies access filters specifying which domains and/or IP addresses are allowed ENS access.

6.4.1.7 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.

6.4.2 domainnotallowed Option

The domainnotallowed option specifies access filters specifying which domains and/or IP addresses are not allowed access for the selected server.

6.4.2.1 Use with the IMAP proxy

The domainnotallowed IMAP Proxy/POP Proxy option (also available at other levels) specifies access filters specifying which domains and/or IP addresses are not allowed access for the selected server.

6.4.2.2 Use with the POP proxy

subsubitem="domainnotallowed MMP/IMAP Proxy option"/>

The domainnotallowed IMAP Proxy/POP Proxy option (also available at other levels) specifies access filters specifying which domains and/or IP addresses are not allowed access for the selected server.

6.4.2.3 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.

6.4.2.4 Use with http

The domainnotallowed MSHTTP option specifies access filters specifying which domains and/or IP addresses are not allowed HTTP access.

6.4.2.5 Use with imap

The domainnotallowed IMAP option specifies access filters specifying which domains and/or IP addresses are not allowed IMAP access.

6.4.2.6 Use with pop

The domainnotallowed POP option specifies access filters specifying which domains and/or IP addresses are not allowed POP access.

6.4.2.7 Use with ens

The domainnotallowed ENS option specifies access filters specifying which domains and/or IP addresses are not allowed ENS access.

Part II Messaging Server command line utilities

Messaging Server comes with a collection of command line utilities. Utilities for the MTA are documented in the MTA command line utilties chapter.

Most Message Store utilities are documented in the "Message Store Command Reference" chapter of the System Administrator's Guide. This includes the following command line utilities: **configutil**, **counterutil**, **deliver**, **hashdir**, **imcheck**, **imdbverify**, **imexpire**, **iminitquota**, **immonitor-access**, **impurge**, **imquotacheck**, **imsasm**, **imsbackup**, **imsconnutil**, **imscripter**, **imsexport**, **imsimport**, **imsrestore**, **mboxutil**, **mkbackupdir**, **msprobe**, **msuserpurge**, **readership**, **reconstruct**, **rehostuser**, **relinker**, **stored**.

Utilities in the `bin` directory are considered public interfaces unless otherwise documented, and utilities in the `lib` directory are considered private interfaces unless they are documented. Output from command line interfaces is considered unstable (subject to change at any time) unless the documentation explicitly states otherwise.

A subset of the Mozilla Network Security Services tools are included in the `lib` directory as a convenience. These include: **certutil**, **modutil**, **pk12util**, **cmsutil**, **ssltap**. The included versions have no functional changes from the Mozilla versions; refer to the Mozilla NSS project for documentation of these utilities.

Chapter 7 configtoxml_utility

7.1 Syntax

```
configtoxml [switches]
```

7.1.1 Restrictions

This is implemented as a Perl script, so a reasonably modern version of Perl must be installed on the system to run this.

7.2 Description

The configtoxml utility converts a legacy configuration to a Unified Configuration.

7.3 Switches

7.3.1 -32|-64

Installation is a 32 bit (-32) or 64 bit (-64) Messaging Server. Default is 64 bit.

7.3.2 --force, -f

Ignore safety checks: allow running as non-root and permit overwriting of any pre-existing Unified Configuration files.

Caution: Using this option may result in a non-functioning configuration. The `restricted.cnf` file must be owned by `root` or the user listed in that file.

7.3.3 --help, -h

Print usage information.

7.3.4 --instance *INSTANCE*, -i *INSTANCE*

Instance name to insert in the generated configuration files. The default is **ims**.

7.3.5 --location *DIR*, -l *DIR*

Read the legacy configuration files from the specified directory DIR. The default is the normal configuration directory.

7.3.6 --noactive, -n

Do not generate an active configuration and do not move the legacy configuration files to *config-root*/legacy-config/. The generated Unified Configuration files will have the names config_.xml, xpass_.xml, and restricted_.cnf and will be written to *config-root*. This may not be used in conjuction with the --output or --undo switches.

7.3.7 --output *CONFIG-FILE PASSWORD-FILE RESTRICTED-FILE*, -o *CONFIG-FILE PASSWORD-FILE RESTRICTED-FILE*

Direct the Unified Configuration file output to the designated files. By default, the files config.xml, xpass.xml, and resricted.cnf are written to the *config-root* directory. This may not be used in conjunction with the --noactive or --undo switches.

7.3.8 --role *ROLE*, -r *ROLE*

Role name to insert in the generated configuration files. The default is to use **ims**.

7.3.9 --yes, -y

Pre-answer any confirmation questions with a "yes" response so that this script can be run without user intervention.

7.3.10 --undo, -u

Remove any active Unified Configuration files and restore any legacy configuration files.

7.4 Usage Notes

Usage Notes: Stop the Messaging Server before running the **configtoxml** command. Alternatively, use the --noactive switch to prevent writing out an active configuration.

When generating an active Unified Configuration, the **configtoxml** command moves all the processed legacy configuration files to the *config-root*/legacy-config directory.

The --undo switch removes the Unified Configuration and restores the legacy configuration files.

7.5 Example

```
# imsimta version
Oracle Communications Messaging Server 7u5-28.12 64bit (built Nov 5 2012)
libimta.so 7u5-28.12 64bit (built 15:58:11, May 23 2012)
Using /opt/sun/comms/messaging/config/imta.cnf (not compiled)
Linux host1.example.com 2.6.39-100.5.1.el5uek #1 SMP Tue Mar 6 20:25:25 EST 2012 x86_64 x86_64 x86_64 GNU/Lin

# configtoxml
WARNING: This procedure will produce an active Unified Configuration which
        will override any existing legacy configuration.

Continue anyway [no]? yes
Creating the directory /opt/sun/comms/messaging/config/legacy-config/
Moving the processed legacy configuration files to /opt/sun/comms/messaging/config/legacy-config/

# bin/imsimta version
Oracle Communications Messaging Server 7u5-28.12 64bit (built May Nov 5 2012)
libimta.so 7u5-28.12 64bit (built 15:58:11, Nov 5 2012)
Using /opt/sun/comms/messaging/config/config.xml (not compiled)
Linux host1.example.com 2.6.39-100.5.1.el5uek #1 SMP Tue Mar 6 20:25:25 EST 2012 x86_64 x86_64 x86_64 GNU/Lin
```

Chapter 8 configure_utility

The configure utility generates an initial product configuration. This is an alias for the init-config utility; see the documentation of that command for details.

Chapter 9 inetuser utility

The inetuser utility is a very limited LDAP provisioning utility for Messaging Server.

9.1 Syntax

```
inetuser --command-file=file
inetuser --help
inetuser --version
inetuser create [switches]user
inetuser show [switches]user
inetuser checkpw [switches]user
inetuser show-domain [switches]domain
inetuser check-dssetup
```

9.1.1 Restrictions

This command uses LDAP configuration settings by default. However, commands that update LDAP generally require Directory Manager credentials and it is a best practice to limit the

access rights available to the administrative account specified by base.ugldapbinddn and by base.ugldapbindcred. As a result, it's typically necessary to specify the `--bind-dn=binddn` and `--bind-pwfile=file` switches to specify a Directory Manager account when updating LDAP directly.

9.2 Parameters

The `create`, `show`, and `checkpw` subcommands take a user identity as a parameter. The user identity is typically the value of the `uid` LDAP attribute (possibly modified by the ldap_uid option) and may include `@domain` to refer to an LDAP domain.

The `show-domain` subcommand takes a domain provisioned in LDAP as a parameter.

No parameters are present when a top-level switch is used or other subcommands are used.

9.3 Description

The inetuser utility is a very limited LDAP provisioning utility for Messaging Server that supports LDAP schema 1 and LDAP schema 2. This tool has been present in Messaging Server for some time and is used by the init-config utility to provision an initial administrative user, group, and associated default domain.

The `create` subcommand is used to create users and domains.

The `show` subcommand is used to show a user's LDAP entry.

The `checkpw` subcommand is used to check a user's LDAP password against the directory. The `inetuser` utility will return a status of `0` if the password is correct.

The `show-domain` subcommand shows a domain's LDAP entry.

The `check-dssetup` subcommand shows information from the `comms_dsseetup` utility that is present in the LDAP directory.

9.4 Examples

The following command creates a user with common name "John Smith" and user identity 'jsmith'. With this command, the email address defaults to 'jsmith@*defaultdomain*' (this assumes the directory manager password is stored in the file *pwfile* in the current directory):

```
# inetuser create -D "cn=Directory Manager" -j pwfile -p "cn=John Smith" jsmith
password:
```

The following command creates a new domain with a new administrative user:

```
# inetuser create -D "cn=Directory Manager" -j pwfile -a all -c newadmin@newdomain.example.com
password:
```

9.5 Switches

9.5.1 --command-file=file, -f file

This top-level switch reads and executes inetuser subcommands from the specified file instead of executing one subcommand from the command line.

9.5.2 --help, -?

This top-level switch displays command usage summary.

9.5.3 --version, -V

This top-level switch displays command version information.

9.5.4 --admin=type, -a type

This create subcommand switch specifies the type of admin user to create. Supported values are `all` (store administrator) and `access` (administrative account used by Messaging Server to authenticate). If not specified, the user account will not have administrative privilege.

9.5.5 --attrlist=attrs, -A attrs

This show subcommand switch specifies a comma-separated list of attributes to show from the user entry, instead of showing all known attributes.

9.5.6 --autocreate, -c

This create subcommand switch will cause the domain to be created when creating a user if it doesn't already exist. Note that the tool requires the first user in a domain to be a store administrator so it's generally necessary to include the `--admin=all` switch with this one.

9.5.7 --bind-dn=binddn, -D binddn

This subcommand switch specifies the bind DN to use for LDAP server authentication. If not specified, the value of the base.ugldapbinddn option is used instead. The credentials specified by that option typically do not have permission to write to the LDAP directory so this switch is usually necessary with the `create` subcommand (as is the `--bind-pwfile` switch).

9.5.8 --bind-pwfile=file, -j file

This subcommand switch specifies a file containing the bind password to use for LDAP server authentication. If not specified, the value of the base.ugldapbindcred option is used as the bind password instead.

9.5.9 --default-domain=domain, -d domain

This subcommand switch specifies the default domain to use if a domain is not explicitly specified. When this switch is not specified, the value of the base.defaultdomain option is used.

9.5.10 --dry-run, -n

This subcommand switch prevents the tool from modifying the LDAP directory. It may be useful to combine this with the `--ldif` switch.

9.5.11 --hostlist=host, -h host

This subcommand switch specifies one or more LDAP server host names to use when connecting to the LDAP server. If not provided, the value of the base.ugldaphost option is used. This may be needed with the `create` subcommand if that option specifies a slave LDAP server rather than a master LDAP server.

9.5.12 --ldapattrval=avl, -p avl

This create subcommand switch specifies an LDAP attribute value list of additional known attributes to include when creating a user. The syntax of the list is *attr1=value1,attr2=value2*. Special characters may be escaped with backslash (\). Alternatively, the value can be base64-encoded by specifying a $ symbol before the equals (=) symbol. The set of known attributes is limited, so if the attribute name is not known by the utility, an error will result.

9.5.13 --ldif=file, -l file

This create subcommand switch specifies a file that will record a copy of the LDIF generated internally by this tool that is used to modify the LDAP directory. Combing this with the `--dry-run` switch is useful to review the changes the tool would make to LDAP. This may also be helpful to customers developing their own provisioning tools.

9.5.14 --logfile=file, -L file

This subcommand switch requests that any diagnostics are appended to the specified file.

9.5.15 --myhost=host, -H host

This subcommand switch specifies the name of the host used to provision store-related attributes such as `mailHost`. If this is not provided, the value of the base.hostname option is used.

9.5.16 --novalidate

Normally the tool will prompt and abort if a mismatch or error is detected. This subcommand switch suppresses that behavior.

9.5.17 --orgdn=dn, -O dn

This create subcommand switch specifies the LDAP DN to use when provisioning a schema 1 organization group in LDAP when creating a domain. This switch is primarily for use by the init-config utility.

9.5.18 --postmaster=mailaddr, -M mailaddr

This create subcommand switch specifies the mail address of the user to include in the postmaster group when creating a domain with a postmaster group. This switch is primarily for use by the init-config utility.

9.5.19 --port=port, -P port

This subcommand switch specifies the LDAP server port to use. If not specified, the value of the base.ugldapport option is used.

9.5.20 --preserveCritical

Normally the tool will prompt and default to overwrite certain critical attributes when performing a create operation and the specified user and/or domain already exists. The subcommand switch prevents the tool from overwriting such attributes.

9.5.21 --pwfile=file, -J file

This create subcommand switch specifies a file containing the password to use when creating a user. If this is not provided, the tool will prompt for a password.

9.5.22 --quiet, -q

This subcommand switch suppresses some prompts and diagnostics.

9.5.23 --require-ssl, -Z

This subcommand switch require use of SSL when communicating with the LDAP server.

9.5.24 --verbose, -v

This subcommand switch requests additional diagnostics from the utility. May be used more than once to increase the amount of diagnostic information.

Chapter 10 init-config utility

The init-config utility initializes the system for use by Messaging Server and generates an initial product configuration.

10.1 Syntax

```
init-config [switches]
```

10.1.1 Restrictions

Must have superuser privileges in order to use this utility. This is implemented as a Perl script, so a reasonably modern version of Perl must be installed on the system to run this. It is more convenient to use this tool if an LDAP server has been installed previously and the `comms_dssetup` tool has been used to initialize the LDAP server.

10.2 Parameters

None.

10.3 Description

The `init-config` tool prepares a system for use by Messaging Server and generates an initial configuration for the product. It can be run in an interactive mode or in a silent mode

(when a state file is provided). By default, the initial configuration generated is designed for product evaluation purposes and enables the primary product services (MTA & Message Store) including a number of common channels and mappings. Alternatively, the tool can be used to generate a limited configuration appropriate as a starting point for a deployment by using the `--recipes=` (or `-r`) switch with a list of initial config recipes (this is new in Messaging Server 8.1). Note that in initial config recipe mode, statefile variables are limited to 1024 characters in length.

The following list summarizes the functions performed by init-config:

1. Creates the Unix user and group for Messaging Server, if needed.
2. If in evaluation or `ldapinit` mode, it creates LDAP server entries (or LDIF) for the default domain, administrative user and postmaster, along with access control lists.
3. If in evaluation or `ldapinit` mode, it creates Directory Server entries (or LDIF) for an administrative group and user for the host where configure is run; this administrative group and user have limited write access to the directory. If in `useldap` mode, this account must already exist and be provided by prompt or state file.
4. Creates DataRoot and ConfigRoot directories with correct permissions.
5. Creates subdirectories of the DataRoot and ConfigRoot directories with correct permissions.
6. Generates a random secret for authentication to local-only services, such as the job_controller and watcher.
7. Prompts you with the minimum number of questions about your environment to create an initial Messaging Server configuration with correct permissions.
8. Attempts to disable mail servers that come with the operating system, for example, Postfix and Sendmail.
9. Prior to MS 8.0.2: runs the imsimta chbuild command to compile character set conversion tables. Starting with MS 8.0.2 a compiled table is bundled in the install package and used unless the customer overrides.
10. Creates convenience symlinks from the install directory to the *DataRoot*, *ConfigRoot*, and *log* directories. Prior to MS 8.0.2, these symlinks were required for correct product functionality.

The following table describes the state file variables used by init-config. State file variables starting with an "ic" prefix are ignored if unrecognized; other state file variables will generate a warning if unrecognized.

Table 10.1 State File variables for initial configuration

Variable Name	Related Configuration	Description
Fqdn.TextField	base.hostname	Fully qualified local hostname. Mandatory.
msg.DataPath	*N/A*	Use a non-default *DataRoot* (not recommended).
iMS.UserId	user in *restricted.cnf*	Specifies the Unix user identity for server processes. Mandatory.
iMS.GroupId	*N/A*	Specifies the primary Unix group of the Unix user if user creation is needed.
UGDIR_URL	*multiple*	An `ldap` or `ldaps` URL for the LDAP server.

UGDIR_BINDDN	*N/A*	Specifies the LDAP administrator user (required for evaluation or ldapinit modes).
UGDIR_BINDPW	*N/A*	Specifies the LDAP administrator password (required for evaluation or ldapinit modes). This password is obfuscated using a ROT-13 variant.
Postmaster.TextField	alias	MTAs require a postmaster for error reports. This option is used directly as an alais value by the mta initial config recipe, or to initialize a Postmaster group by evaluation or ldapinit modes.
admin.password	base.proxyadminpass, etc.	Password for the store administrative user primarily used for proxy login between servers. This password is base64-encoded for obfuscation.
EmailDomain.TextField	base.defaultdomain	Default email domain.
OrgName.TextField	*N/A*	Default organization DN, only used by evaluation or ldapinit modes.
InternalIPlist	INTERNAL_IP mapping table	Specifies IP address information, in comma-delimited form, for systems permitted to relay mail without authentication. Note: when used in initial config recipe mode, all statefile variables have a length limit of 1024 characters. As a result, additional statefile variables `ic.iplist2`, `ic.iplist3`, ... can be used to provide additional values for this setting.
ic.iplist2, ic.iplist*n*	INTERNAL_IP mapping table	In initial config recipe mode, this specifies IP addresses for systems permitted to relay in addition to the ones in the `InternalIPlist`. Each of these values is limited to 1024 characters. These are read in order, starting with InternalIPlist, then ic.iplist2, ic.iplist3, ic.iplist4, ... stopping when no value is found in the statefile. The values are sorted before producing the INTERNAL_IP mapping table.
XMLCONFIG	*N/A*	Starting in Messaging Server 8.1, this option is ignored and Unified Configuration is always used. For previous versions, when set to 1 this creates a Unified Configuration and when set to 0, creates a legacy configuration. The --xml and --noxml command options override what is specified in the statefile.
cassandra	store.dbtype, isc.enable	When set to 1, use a Cassandra message store rather than a classic message store. New in Messaging Server 8.0.2.
dssetup.ugsuffix	base.ugldapbasedn	Specifies the user/group suffix. Unless `--noldap` is specified, the default comes from LDAP `cn=CommsServers,o=comms-config`.
dssetup.dcsuffix	base.dcroot	Specifies the domain suffix. Unless `--noldap` is specified, the default comes from LDAP `cn=CommsServers,o=comms-config`.

dssetup.schematype	base.ldap_schemalevel	Specifies the schema type. Unless `--noldap` is specified, the default comes from LDAP `cn=CommsServers,o=comms-config`.
admin.user	store.admins, base.proxyadmin	Administrative user for proxy authentication and store administration. Defaults to "admin". New in Messaging Server 8.0.2.3.
ugldap.hostlist	base.ugldaphost	LDAP server (a space-delimited failover server is permitted). When present, this overrides UGDIR_URL. New in Messaging Server 8.0.2.3.
ugldap.port	base.ugldapport	LDAP server default port. When present, this overrides UGDIR_URL. New in Messaging Server 8.0.2.3.
ugldap.usessl	base.ugldapusessl	When to require SSL for LDAP. When present, this overrides UGDIR_URL. New in Messaging Server 8.0.2.3.
ugldap.binddn	base.ugldapbinddn	LDAP administrative user for server operations. This user should be able to read all user entries but should have limited write access to the directory. Such a user is created by evaluation or ldapinit modes. New in Messaging Server 8.0.2.3.
ugldap.bindcred	base.ugldapbindcred	Password for server operations. This password is base64-encoded for obfuscation. New in Messaging Server 8.0.2.3.
ic_lmtp_ipmask_list	LMTP_ACCESS mapping	Used only by `lmtpserver` initial config recipe. This specifies IP address information, in comma-delimited form, for systems permitted to deliver mail to the LMTP server without authentication. The LMTP_ACCESS mapping table works the same way as the INTERNAL_IP mapping table. New in Messaging Server 8.1.
icmmpimap_enable	mmp.enable, *etc.*	Used only by `proxy` initial config recipe. When set to 1, this enables the MMP and settings for an MMP IMAP proxy. New in Messaging Server 8.1.
icmmppop_enable	mmp.enable, *etc.*	Used only by `proxy` initial config recipe. When set to 1, this enables the MMP and settings for an MMP POP proxy. New in Messaging Server 8.1.
icmshttpd_enable	http.enable, *etc.*	Used only by `proxy` initial config recipe. When set to 1, this enables the Convergence mshttpd proxy and settings for submission proxy via mshttpd. New in Messaging Server 8.1.
icsubmithost	alarm.noticehost, http.smtphost	Required by the `proxy` initial config recipe. This specifies the hostname of a submission server used for mail from the alarm subsystem and Convergence mshttpd (if enabled). New in Messaging Server 8.1.
icsubmitport	http.smtpport	Used only by `proxy` initial config recipe. This specifies the port of a submission server used for mail from Convergence mshttpd (if enabled).

		This defaults to 587; however use of 465 for SSL-encrypted submission should be considered. New in Messaging Server 8.1.

10.4 Switches

10.4.1 --cassandra

Configure use of the Cassandra Message Store rather than classic Message Store.

10.4.2 --dataroot=*dataroot*

Specifies an alternate location for the data root directory where read/write product configuration and operational data will be stored. The default location for the data root directory is `/var/`*`server-root`* where *`server-root`* is the installation directory. Use of a non-default *`dataroot`* is not recommended.

10.4.3 --debug

Requests additional debugging information while generating initial configuration; this primarily impacts LDAP operations.

10.4.4 --help, -?

Display command usage summary.

10.4.5 --ignoreSendmail

Ignore the presence of Sendmail or Postfix on the system. By default, the initial configuration tool attempts to avoid port conflicts by disabling Sendmail or Postfix if those tools are enabled.

10.4.6 --ldapport=*ldapPort*

Use a non-default LDAP port to communicate with the LDAP server.

10.4.7 --ldif

Treat the LDAP directory as read-only and generate an LDIF file in a *DataRoot*/install/configure.ldif file for any desired changes to the directory. The administrator must apply the LDIF file to the directory after initial configuration but before starting the Messaging Server product. This is helpful if the person doing the Messaging Server installation does not have directory administrative rights.

10.4.8 --list-recipes

Displays the available initial configuration recipes and exits.

10.4.9 --noldap

Create a full evaluation installation without an LDAP server present. This is used in combination with the `--state=`*statefile* and `--ldif` switches to configure Messaging Server

when the LDAP server is not available at the time initial configuration is generated. Normally the answers provided to `comms_dssetup` questions are read from the LDAP server by this tool; but when this switch is used, those values should be provided in the state file. Note that it's possible to both configure and use a routing-only MTA without LDAP; use of the `--recipes` switch (new in Messaging Server 8.1) is the recommended way of doing that.

10.4.10 --novalidate

Skip most validation of user input and state file variables. This may result in a non-functional configuration. This is provided primarily as a mechanism to work around possible bugs in the init-config tool's validation logic.

10.4.11 --noxml

Generate an initial legacy configuration rather than an initial Unified Configuration (XML based). This option is not supported starting with Messaging Server 8.0.2, and no longer works starting with Messaging Server 8.1.

10.4.12 --preserveCritical

When initializing the LDAP server for use by Messaging Server, this prevents certain critical attributes from being overwritten with different values, even if new values are provided. See the installation guide for a more in-depth discussion of critical LDAP attributes.

10.4.13 --quiet, -q

Suppress most non-error console output (a logfile is still created containing necessary information to diagnose issues).

10.4.14 --recipes=*recipe-list*, -r *recipe-list*

Generate a minimal configuration for a particular deployment function as determined by a list of initial configuration recipes. LDAP will not be involved unless `ldapinit` or `useldap` is included in the recipe list.

10.4.15 --saveState=*statefile*

Specifies the location to save a state file. The default location is *DataRoot*`/setup/saveState`.

10.4.16 --ssl

Require SSL when communicating with the LDAP server. Any necessary SSL certificates and configuration must be manually set up prior to using this option.

10.4.17 --state=*statefile*

Use a silent installation file and do not prompt the administrator for information.

10.4.18 --version, -V

Displays the product version and exits.

10.4.19 --xml

Force use of Unified Configuration, even if the statefile specifies use of legacy configuration. Starting with Messaging Server 8.1, this is the default behavior so this switch has no effect.

10.5 Examples

To configure an evaluation installation of Messaging Server when an LDAP server is present (interactive).

```
# init-config
```

To initialize LDAP server and configure the first classic Message Store in a multi-system deployment (interactive).

```
# init-config -r ldapinit,store,lmtpserver
```

To configure an additional classic Message Store in a multi-system deployment (interactive).

```
# init-config -r useldap,store,lmtpserver
```

To add an MTA with submission service to a Messaging Server multi-system deployment (interactive).

```
# init-config -r useldap,mta,submit
```

To add an MMP and/or mshttpd service to a Messaging Server multi-system deployment (interactive).

```
# init-config -r useldap,proxy
```

To configurate a routing-only MTA without LDAP (interactive).

```
# init-config -r mta
```

To configurate using a state file (non-interactive).

```
# init-config --state=/path/to/statefile
```

To list available initial configuration recipes.

```
# init-config --list-recipes
Available Initial Configuration Recipes:
 ldapinit        Initialize LDAP directory for use by Messaging Server
 lmtpserver      LMTP server (used by initial configuration)
 mta             Configure an MTA (used by initial configuration)
 none            Minimal configuration with external services disabled
```

```
proxy           Proxy configuration (MMP, Convergence mshttpd)
store           Configure a message store (used by initial configuration)
submit          Add submission service to an MTA (initial configuration)
useldap         Enable support for LDAP users (doesn't initialize LDAP)
```

Chapter 11 msconfig utility

11.1 Syntax

```
msconfig [switches] [command] [\command]...
```

11.2 Parameters

The *msconfig* utility may be used interactively by invoking it without any commands:

```
msconfig
```

or noninteractively by specifying one or more commands as an argument, e.g.,

```
msconfig SET <option> <value>
```

11.2.1 Switches

The *msconfig* utility accepts a number of switches on the invocation line.

11.2.1.1 -directory=*config-directory*

Specifies an alternate location for Messaging Server configuration files. msconfig will read and write config.xml and xpass.xml from this alternate location. The directory where the files are located must be given as an argument.

11.2.1.2 -help

Prints basic information about using msconfig and enters the msconfig help system. msconfig exits after help system access is terminated.

11.2.1.3 -input=*input-file*

If specified, msconfig will read commands from a file instead of from the terminal. The input file name must be given as an argument. If -input is specified no msconfig commands can be given on the invocation line.

11.2.1.4 -multiple

If specified, the -multiple switch allows multipe commands to be specified on the command line separated by backslashes "\".

11.2.1.5 -novalidate

Read the configuration but do not validate it. Warning: This should only be used in extreme situations where a broken configuration needs to be analyzed. Such a configuration should NEVER be used in production.

11.2.1.6 -output=*output-file*

If specified, msconfig will direct its output to the specified file.

11.2.1.7 -page

Controls whether or not msconfig help system output stops and prompts after each page. -page is the default; use -nopage to disable output paging.

11.2.1.8 -prompt

Controls whether or not the help system prompts for additional topics after a topic has been output. -prompt is the default; use -noprompt to turn off help system prompting.

11.2.1.9 -readonly

Open configuration in readonly mode and without locking. This is useful when performing display operations that should not fail when someone is editing the configuration.

11.2.1.10 -remark=*remark-string*

Specifies a remark to attach to configuration as it is written. The remark string must be given as an argument.

The primary use of -remark is to specify a remark when a command is given on the invocation line, e.g.,

```
msconfig -remark "Enable enqueue debugging" set mm_debug 5
```

11.2.1.11 -require=*conditions*

The -require switch specifies a list of conditions that must be met for "successful" operation. msconfig will exit with a nonzero status if these conditions are not met. The available conditions are:

- write - One or more configuration writes must have been performed.

11.2.2 Commands

List of *msconfig* utility commands:

11.2.2.1 DEFAULT

Reset the location where options are set to default location.

11.2.2.2 DEPLOYMAP operation

Manipulate deployment maps. The supported operations are ADD, DELETE, DUMP, LIST, CREATE, RENAME, SET, READ, and WRITE.

11.2.2.3 DIRECTORY [filter]

Displays the recipes whose names match the specified <filter>.

11.2.2.4 DIFFERENCES [m [n]]

Compare configurations.

11.2.2.5 EDIT object

Places the specified object in a file and invokes the editor to enable editing.

11.2.2.6 EXECUTE string

Execute the specified string as if it were a one-line recipe. The final value computed by the recipe is printed at the end of execution.

11.2.2.7 EXIT

Exit msconfig utility with a prompt to write if configuration was modified.

11.2.2.8 HELP [topic[subtopic...]]

Display help.

11.2.2.9 HISTORY

Display previous saved configurations.

11.2.2.10 INSTANCE

Store options in instance.

11.2.2.11 IMPORT config

Read configuration from alternate file(s).

11.2.2.12 LOG

Display previous saved configurations. Synonym for `HISTORY`

11.2.2.13 QUIT

Exit *msconfig* utility discarding changes.

11.2.2.14 REVERT [n]

Discard modifications and reload configuration.

11.2.2.15 ROLE

Store options in role.

11.2.2.16 RUN recipe [arg1 [arg2....]]

Run the specified recipe file with the specified arguments. If no file path is given as part of "recipe" the command will look for the file in <configroot>/recipes, and if not found there, it will look in <serverroot>/lib/recipes.

The arguments specified in the run command can be accessed in the recipe with the argc and argv functions.

11.2.2.17 SET option [value1 [value2....]]

Set option to the specified value(s).

11.2.2.18 SHOW option [namefilter [valuefilter]]

Show value of option.

11.2.2.19 UNSET option

Delete option setting.

11.2.2.20 WRITE

Write out configuration changes.

11.3 Description

The *msconfig* utility is used to examine and modify Messaging Server configuration. See Configuration syntax for more information on the Messaging Server configuration.

The utility performs type checking on configuration settings it makes. See Option value syntax for further details.

11.4 Prompts

When run interactively, msconfig's default prompt is:

```
msconfig>
```

When modifications have been made to the configuration the prompt will change to:

```
msconfig#
```

The DEFAULT, INSTANCE, and ROLE commands also affect the prompt:

- Default mode:

```
msconfig>
```

- Instance mode:

```
msconfig.instance>
```

- Role mode:

```
msconfig.role>
```

11.5 Return status

msconfig returns a status of 0 if the last command executed successfully, 1 if the last command generated an error, and 2 if the last command could not be performed because the configuration was locked.

msconfig will also return a status of 1 if the conditions specified as arguments to the -require switch are not met.

11.6 Some Useful Commands

11.6.1 Help Command

The *msconfig* utility is a comprehensive help utility and its help command can be used to get help not only on the utility and its commands but also on various topics pertaining to the Messaging Server.

```
msconfig help
```

This lists all the Topics on which help is available. Choose a specific Topic which gives some information and then one can enter additional sub-topics for more in depth information.

Help on specific msconfig commands can be accessed in interactive mode by entering:

```
help commands            <- List of available commands
help commands <command>  <- Information about command <command>
```

Help on the options msconfig can manipulate can be accessed by entering:

```
help option <option>     <- Information about option <option>
```

Note that options in the help system should be specified without any associated scope.

Use the `-search` switch along with list of strings to display ALL articles that contain the specified strings.

```
help -search string1 [string2...]
```

Use the `-keyword` switch along with list of keywords to display ALL articles marked with any of the specified keywords.

To disable the prompts when additional information is available, use the `-noprompt` switch.

Help output by default is paged. Use `<space>` to display additional pages. `<return>` displays an additional line. `a` displays remaining information without paging and `q` stops the output. Use `-nopage` switch to disable paging.

11.6.2 Show Command

Use the *msconfig show* command to display current settings.

For example, to show all currently enabled options:

```
msconfig show *enable
```

11.6.3 Set Command

Use the *msconfig set option [value1 [value2]]* command to set option values.

Use the *-prompt* switch for entering passwords since it causes the utility to prompt for the value without echoing it.

Values must be quoted if they contain spaces. In non-interactive mode, each special character must be prefixed with the escape character "\". For example, to set the value of the *auth.searchfilter* option to *(|(uid=%U)(mail=%o)):*

```
msconfig set auth.searchfilter \"\(\|\(uid\=\%U\)\(mail\=\%o\)\)\"
```

11.6.4 Edit Command

Use the *msconfig editobject* command to edit the specified object in a file in an appropriate textual form. The *msconfig edit* command invokes the editor specified by the EDITOR shell variable. You can then make the change, save it, exit, and your configuration is updated.

For example, suppose you want to set the master_debug option on the tcp_local channel:

```
msconfig set channel:tcp_local.master_debug
```

Even if you did not know the preceding command, you could still use the msconfig command to perform this operation by invoking it in edit mode:

```
msconfig edit channels
```

You can also edit a single channel block by itself by running the following command:

```
msconfig edit channel tcp_local
```

Channel-specific option files are mapped into the Unified Configuration as sub-elements of a general "options" channel option. These options appear at the bottom of each channel block when you run edit channels. The following example illustrates this point:

```
tcp_local identnonenumeric inner loopcheck maysaslserver maytlsserver mx \
 pool SMTP_POOL remotehost saslswitchchannel tcp_auth smtp sourcespamfilter1 \
 switchchannel
tcp_local-daemon
==
trace_level=2
```

The *msconfig* command provides the same editing capability for MTA rewrite rules *(edit rewrites)*, mappings *(edit mappings)*, MTA conversion channel control entries *(edit conversions)*, Sieve filters *(edit filter)*, options *(edit option)* and MTA aliases *(edit aliases)*

All objects placed in the file for editing are deleted prior to reloading the file's contents. Use `-noreplace` to treat the resulting file contents as configuration additions.

All configuration changes are normalized and validated by default. Use `-novalidate` to disable validation.

11.6.5 List and Compare Configurations Commands

The repository of previous configurations, known as the graveyard, is stored in the *ConfigRoot/old-configs/* directory. The move from current configuration to the graveyard is performed when a new configuration is written to disk. The graveyard maintains the most recent 100 configurations. With the graveyard, you can restore an old configuration by reverting to a previous configuration. Furthermore, you can compare differences between any two configurations, for example, between the active configuration and a previous configuration, or two old configurations.

```
msconfig history         <- List of configurations in the graveyard

msconfig differences     <- Compare configurations
```

Chapter 12 refresh utility

The refresh utility refreshes the configuration of running Messaging Server components.

12.1 Syntax

```
refresh [components ...]
```

12.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility.

12.2 Parameters

12.2.1 *component*

One or more *component* parameters may be supplied to refresh specific running components. See start-msg utility for a list of the valid values for *component*.

12.3 Description

This triggers a configuration refresh of one or more running Messaging Server processes. The actual configuration refresh by each process happens asynchronously.

Note: The MTA's design makes it easy to restart a dispatcher server process without service impact. In general, therefore, MTA options are not refreshable (with mappings being a notable exception). For example, `imsimta restart smtp` causes the dispatcher to start new SMTP server processes for future incoming connections, reading a new configuration. This happens without service interruption. The Message Store, ENS, and MMP design makes restarting a process a service outage, so it is important for refresh logic to be built for most options in these servers. Most of these options are therefore refreshable.

Having msconfig print a restart required warning would generate a lot of pointless warnings for MTA options which is undesirable behavior for Unified Configuration. Because of this, the msconfig utility does not issue a warning when an option that requires a restart is modified. The legacy configutil tool does issue such a warning.

12.4 Example

The following command refreshes the IMAP server:

```
# refresh imap
```

Chapter 13 start-msg utility

The start-msg utility starts enabled Messaging Server components.

13.1 Syntax

```
start-msg [-m] [-l] [-L] [components ...]
```

13.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility. Only one start-msg or stop-msg process may run at a time on the system.

13.2 Parameters

13.2.1 ha

The `ha` parameter starts the server in high-availability mode and is normally only used by a cluster agent control script.

13.2.2 *component*

One or more *component* parameters may be supplied to start a specific enabled component. Valid values for *component* include:

- `watcher` - watches for process failure and triggers restart of appropriate components.
- `ens` - event notification server (enpd); required for IMAP IDLE.
- `metermaid` - legacy RAM-based storage facility similar to memcache or Redis; see MeterMaid documentation.
- `store` - stored message store maintenance process.
- `purge` - impurge message store daemon.

- `imap` - IMAP server.
- `pop` - POP3 server.
- `isc` - Indexed search converter service (required by Elasticsearch feature).
- `cert` - Certificate validation daemon used by Convergence S/MIME feature.
- `http` - Convergence mshttpd proxy.
- `sched` - Scheduler daemon for period tasks.
- `deploymap` - feature in development.
- `dispatcher` - Multi-process service controller used by the MTA and LMTP server (starts other processes).
- `job_controller` - MTA's outbound delivery controller (starts other processes).
- `snmp` - SNMP monitoring sub-agent.
- `sms` - SMS gateway service.
- `mmp` - Messaging Multiplexor (POP and IMAP proxy).
- `rollovermanager` - Time-based log-file rollover service.
- `mta` - an alias for the dispatcher and job_controller.

13.3 Description

This starts one or more enabled Messaging Server components. Which components are started can be controlled by using the appropriate enable option, e.g., `store.enable` to enable the store, or the appropriate enablesslport option the appropriate component's scope., e.g., `imap.enablesslport` for IMAP's SSL port. If a specified component is already running, the component will continue running unchanged. If a specified component is not enabled, an error will be displayed for that component.

13.4 Switches

13.4.1 `-l`

List active servers and exit.

13.4.2 `-L`

List enabled servers and exit.

13.4.3 `-m`

Start the message stored process as the preferred replication master when using classic Message Store with automatic failover and DB replication enabled. This should be used once when setting up a replication affinity group.

13.5 Examples

The following command starts all enabled Messaging Server components:

```
# start-msg
```

The following command starts the **imap** server process(es):

```
# start-msg imap
```

Chapter 14 stop-msg utility

The stop-msg utility stops running Messaging Server components.

14.1 Syntax

```
stop-msg [-f] [components ...]
```

14.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility. Only one start-msg or stop-msg process may run at a time on the system.

14.2 Parameters

14.2.1 ha

The ha parameter stops the server in high-availability mode and is normally only used by a cluster agent control script.

14.2.2 *component*

One or more *component* parameters may be supplied to stop a specific running component. See start-msg utility for a list of the valid values for *component*.

14.3 Description

This stops one or more running Messaging Server components. Stopping components generally results in a service outage for that component. If the goal is to activate changes in refreshable product options, see the refresh utility and the imsimta reload utility.

14.4 Switches

14.4.1 -f

Force stop using SIGKILL (may result in metadata corruption in the classic Message Store).

14.5 Examples

The following command stops all running Messaging Server components:

```
# stop-msg
```

The following command stops the **impurge** server process:

```
# start-msg purge
```

Part III 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. (A few special, security-related options establishing the main Messaging Server user id, *etc.*, are not set as `config.xml` options, but instead are stored in the `restricted.cnf` file.)

Chapter 15 `restricted.cnf` file

As of MS 7.0.5, certain special, security-related "options" -- defining fundamental Messaging Server Unix accounts -- are stored in the `restricted.cnf` file (for both Unified Configuration and legacy configuration). (Note that these few "options" are not settable using `msconfig`; instead edit the `restricted.cnf` file.) These "options" include:

1. `user`, replacing as of MS 7.0.5 the former `imta_user` MTA Tailor option (specifying the UNIX user id of the MTA user) and the former `serveruid` Message Store option (specifying the UNIX user id of the Message Store user).

2. `group`, replacing as of MS 7.0.5 the former `imta_world_group` MTA Tailor option (specifying the UNIX group id of the MTA user). Note that the `servergid` Message Store option (specifying the UNIX group id of the Message Store user) was deleted in MS 7.0.4; the Message Store now always uses the primary group of the user specified via the `serveruid` Message Store option.

3. `pipeuser`, as of MS 8.0 preferred to the deprecated use of the `user` channel option on the pipe channel (specifying the username under which the channel runs).

4. `allow_pipe_setuid`, new in MS 8.0; when set to `0` (the default), the pipe channel is only allowed to run as the specified `pipeuser`; if set to `1`, then the pipe channel is allowed to run as other users.

5. `noprivuser`, replacing as of MS 7.0.5 the former `imta_user_username` MTA Tailor option (specifying the UNIX user id for certain untrusted operations such as MTA mapping table sequence number file access, or untrusted pipe channel operations).

Chapter 16 Base options

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.

16.1 accounturl Option

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

16.2 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.

16.3 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.

16.4 bgmax Option

The bgmax option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See bgpenalty for more information.

16.5 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.

The bgpenalty option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the length of time in seconds added to the authentication delay after each failed authentication.

16.6 bgmaxbadness Option

The bgmaxbadness option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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.

16.7 bgdecay Option

The bgdecay option (available under base, imap, pop, mmp, imapproxy, and popproxy) represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See bgpenalty for more information.

16.8 bglinear Option

The bglinear option (available under base, imap, pop, mmp, imapproxy, and popproxy) defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See bgpenalty for more information.

16.9 bgexcluded Option

The bgexcluded option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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).

16.10 dblockcount Option

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

16.11 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.

16.12 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.

16.13 debugkeys Option Under base

The debugkeys base option specifies a space-separated list of keywords used to enable various optional debugging facilities; see Keywords That May Be Included in debugkeys Option Value.

Note that the SMTP server's AUTH_DEBUG TCP/IP-channel-specific option can override debugkeys for SMTP server authentication purposes.

For Message Store and other non-MTA processes, setting a relevant debugkey will enable NOTICE-level logging in the logfile for that process. The MTA has a different logging model and requires two additional settings to see debugging associated with a debugkey. First, MTA debug log files must be enabled (via master_debug, slave_debug, or the equivalent finer-grained mechanism). Second, it's necessary to set mta.mm_debug to a value of at least 3 for the DKIM-related debugkeys or to set mta.os_debug to 1 for the LDAP and authentication-related debugkeys.

16.14 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).

16.15 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 MTA's SMTP server and DNS reverse lookups, see the ident* channel options.

16.16 enablelastaccess Option

The enablelastaccess base option enables last access time tracking. Access time data is used by imsconnutil and mboxutil -o -t.

16.17 filterurl Option

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

16.18 folderurl Option

The folderurl base option specifies the URL for personal folder management (obsolete).

16.19 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.

16.20 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.

16.21 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.

16.22 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.

16.23 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.

16.24 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 16.1 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

16.25 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.

16.26 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.

16.27 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.* building and requesting a list of "known" domain attributes. The default of 0 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.

The "known" attribute list consists of all attributes specified in the various ldap_domain_attr_* and similar MTA options, as well as any attributes specified in any LDAP domain map attribute substitutions. This should cover all the cases where the MTA requests domain attributes via its internal domain lookup facilities. However, in the unlikely event that other calls are made to the domain map facility, a site that has added additional

domain attributes may be forced to use the default setting of 0 so that LDAP domain queries will return those additional, custom LDAP attribute values.

16.28 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.

16.29 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).

16.30 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.

16.31 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.

16.32 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 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.

(Note that the acceptalladdresses channel option, if used, modifies the timing and form of the rejection.)

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

(New in 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.

16.34 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)
```

16.35 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 Message 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.

16.36 ldapconnecttimeout Option

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

16.37 ldapmodifytimeout Option

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

16.38 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.

16.39 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.

16.40 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.

16.41 ldaptrace Option

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

16.42 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.

16.43 listurl Option

The listurl base option specifies the URL for mailing list management (obsolete).

16.44 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.

16.45 loginseparator Option

The loginseparator base option specifies the character(s) to be used as login separator (between userid and domain).

16.46 obsoleteimap Option

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

16.47 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.

16.48 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.

16.49 properties Option

The properties base option and rolename option are used only during initial configuration by a Deployment Map client to convey information to a remote Deployment Map server.

16.50 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.

16.51 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.

16.52 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).

16.53 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.

16.54 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.

Note that if the proxytrustmailhost Base option is enabled, then a user's mailHost LDAP attribute value will be "trusted" for proxying purposes as if it had been specified in the proxyserverlist.

16.55 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.

16.56 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). (If this option is not set, the IMAP server will also check the accounturl base option.)

Note that a trailing "/" is required on the URL, e.g.,

```
msconfig set base.pwchangeurl http://webmail.example.com/
```

See also the IMAP password expiration alert options.

16.57 rbac Option

The rbac base option enables use of Role-Based Access Controls on Solaris (don't require root access).

16.58 rfc822headerallow8bit Option

The rfc822headerallow8bit base option, if set to 1, allows 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 "?".

16.59 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.

16.60 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.

16.61 sitelanguage Option

The sitelanguage base option specifies the default language tag for use by the Message Store (and its command line utilities), and the MSHTTP server.

16.62 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.

16.63 ssladjustciphersuites Option

The ssladjustciphersuites option allows adjusting which SSL cipher suites are enabled or disabled. (This option is available under base, mmp, imapproxy, popproxy, and vdomain.) 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 crytographic 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 will 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 cipher suites that require installation of special server certificate types, for example the DSS cipher suites. Such a site would enable these additional suites once installation was complete.

3. If a site is forced to continue supporting a particularly old client that only supports old cipher suites, they can be explicitly enabled (for example '+RC4' enables the RC4 cipher suites).

4. A site that chooses to disable an older cipher or hash function pro-actively despite potential interoperability issues may choose to do so. For example, to disable all ciphers using the '3DES' or 'SHA1' algorithms, simply set '-3DES,-SHA1'. Be aware that this sort of pro-active action may generate support calls from end users running older mail clients.

5. 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 '3DES' algorithm, simply set '-3DES'.

As of NSS 3.28 (2017), the following cipher suites are enabled by default in the NSS library: TLS_AES_128_GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256, TLS_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256, TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_3DES_EDE_CBC_SHA.

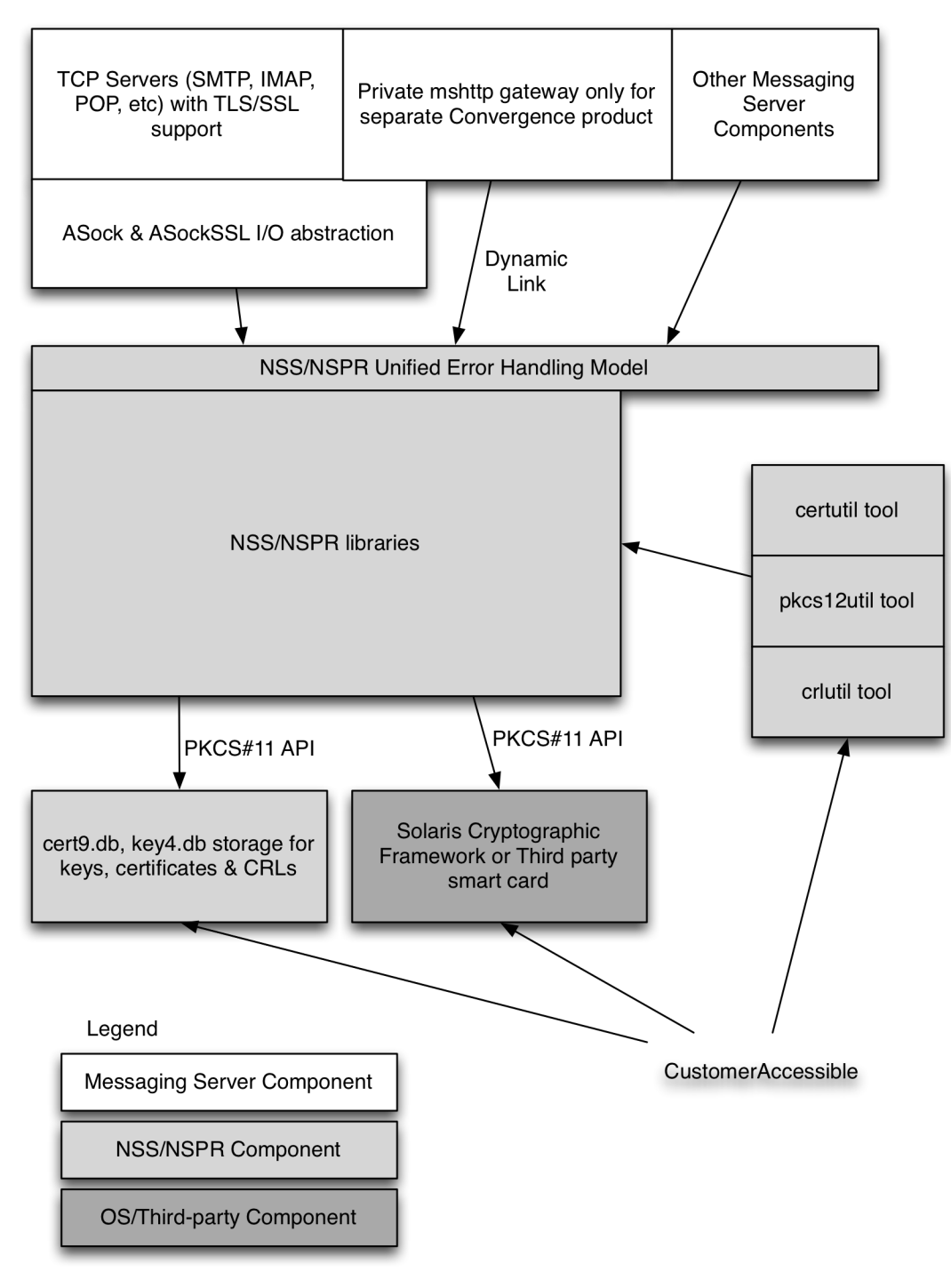
The complete list of cipher suites present in NSS 3.28 (2017) follows: SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA, SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA, SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5, SSL_RSA_EXPORT_WITH_RC4_40_MD5, SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA, SSL_RSA_FIPS_WITH_DES_CBC_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_RSA_WITH_DES_CBC_SHA, SSL_RSA_WITH_NULL_MD5, SSL_RSA_WITH_NULL_SHA, SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA, TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_256_CBC_SHA, TLS_DHE_DSS_WITH_RC4_128_SHA, TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,

TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_NULL_SHA, TLS_ECDHE_ECDSA_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_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_RSA_WITH_NULL_SHA, TLS_ECDHE_RSA_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_ECDH_ECDSA_WITH_NULL_SHA,
TLS_ECDH_ECDSA_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_ECDH_RSA_WITH_NULL_SHA, TLS_ECDH_RSA_WITH_RC4_128_SHA,
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA,
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256,
TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384,
TLS_RSA_WITH_NULL_SHA256, TLS_AES_128_GCM_SHA256,
TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256,
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256, TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384,
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA,
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA,
TLS_DHE_DSS_WITH_RC4_128_SHA, TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA,
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA,
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_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_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_RSA_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_ECDH_ECDSA_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_ECDH_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256,
TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384,
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA, TLS_RSA_WITH_CAMELLIA_256_CBC_SHA,
TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_RC4_128_SHA,
TLS_RSA_WITH_SEED_CBC_SHA.

TLS 1.3 (NSS 3.39+) has its own cipher suites separate from previous TLS versions. These include: TLS_AES_128_GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256, TLS_AES_256_GCM_SHA384.

Always keep in mind that adjusting the available ciphersuites can impact multiple protocols; NSS is tightly integrated into Messaging Server, as the following diagram illustrates:



16.64 sslcachedir Option

The sslcachedir option, (available under base, mmp, imapproxy, and popproxy), 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 predictable, this option must be the same for the IMAP Proxy and POP Proxy -- any settings of this option for the proxies must match. (Or better yet, don't set it explicitly for any of the proxies; instead set it at MMP level.)

16.65 ssldbpath Option

The ssldbpath base option specifies the location of certificate and key files. This defaults to the product configuration directory; see the CONFIGROOT environment variable.

Additional base options impacting the certificate and key file names and format are ssldbprefix and (prior to MS 8.0) ssldblegacy. Prior to MS 7.0.5.31.0, see also the sslcachedir option.

16.66 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.

Additional base options impacting the certificate and key file location and names are ssldbpath and ssldbprefix.

16.67 ssldbprefix Option

The ssldbprefix base option specifies the prefixes of the certificate and key files.

Additional base options impacting the certificate and key file location and format are ssldbpath and (prior to MS 8.0) ssldblegacy.

16.68 sslcompress Option

The sslcompress base option determines whether support for the SSL/TLS Compression option (RFC 3749) is enabled. Enabling this is not recommended.

16.69 sslnicknames Option Under base

The sslnicknames Base option specifies a list of the nicknames of the certificates in the SSL certificate database to offer as the server certificate if SSL/TLS is enabled. Only one nickname of each certificate type is permitted (*e.g.*, one RSA certificate, one DSS certificate) so normally only one will be specified. The default is "Server-Cert". This base level list may be overridden for particular services.

16.70 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.

16.71 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.

16.72 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.

16.73 sslconnlimit Option

The sslconnlimit boolean option determines when SSL connections will be rejected if a connection limit specified by connlimits or a DNS RBL blocklist specified by dnsrbl is encountered. By default, the connection is rejected after SSL is negotiated using an application-level protocol error describing the rejection. If this is set, the connection is rejected prior to SSL negotiation by sending a fatal SSL user_canceled alert at the SSL protocol layer.

16.74 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.

16.75 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 stress fd wait is a new feature, it may be helpful to disable the feature temporarily if it is causing an unexpected problem.

16.76 supportedlanguages Option

The `supportedlanguages` base option specifies the languages supported by server code.

16.77 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.

16.78 tlsminversion Option

16.78.1 Use with base

The `tlsminversion` base option determines the minimum acceptable version of TLS (the modern version of the SSL protocol). This presently takes a value of `TLS1.0`, `TLS1.1`, `TLS1.2`, and new in Messaging Server 8.1: `TLS1.3`. This option defaults to `TLS1.2` (disabling TLS 1.0 & 1.1 by default). Prior to the 8.1 release, TLS 1.1 was the default and prior to the 8.0.1 release, TLS 1.0 was the default. If this is set to `TLS1.2` (the current default), the setting of the `tlsv12enable` option is ignored. If this is set to `TLS1.3`, the setting of the `tlsv13enable` option is ignored.

16.78.2 Use with channel

When placed on a channel, the `tlsminversion` option overrides the base default for the minimum TLS version for all messages dequeued from the channel.

16.79 tlsv12enable Option

The `tlsv12enable` base option determines whether TLS version 1.2 (the recommended 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 `TLS1.2` or `TLS1.3`, this option is ignored and TLS 1.2 is enabled.

16.80 tlsv13enable Option

The `tlsv13enable` base option determines whether TLS version 1.3 (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. If `tlsminversion` is set to `TLS1.3`, this option is ignored and TLS 1.3 is enabled.

16.81 tmpdir Option Under base

The tmpdir base option specifies the temporary file directory; defaults to $DATAROOT/tmp/.

On Linux, this option should instead be set to /dev/shm/.

16.82 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.

16.83 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.

16.84 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.

16.85 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.

16.86 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.

16.87 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.

16.88 welcomemsg Option Under base

The welcomemsg Base option specifies a default welcome message for new users of the Message Store. The maximum size is 1 MB.

Syntax: "$" line separators, with headers.

Note that the base value can be localized using the welcomemsg option set under named message_language groups.

This default welcome message message specified at base level is only used if there is no domain-specific welcome message in a preferred language (no mailDomainWelcomeMessage tagged with a preferred language) set on the LDAP entry of the domain in which the new user resides.

16.89 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, imapproxy, popproxy, messagetrace, pop, snmp, rollovermanager, tcp_lmtp_server, transactlog, 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.

16.89.1 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).

16.89.2 flushinterval Option

The flushinterval logfile option, *component*.logfile.flushinterval, specifies the time interval in seconds between logfile buffer flushes.

16.89.3 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.

16.89.4 `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.

16.89.5 `loglevel` Option Under logfile

The `loglevel logfile` option specifies the logging level for the whatever component's logfile it is set under. *E.g.*, `imap.logfile.loglevel` specifies the logging level for the IMAP server's log file.

Valid values for the `loglevel` option are: `nolog`, `emergency`, `alert`, `critical`, `error`, `warning`, `notice`, `information`, or `debug`. The default is `notice`.

The `loglevel logfile` option for the MTA specifies an MTA log level used for the `imta` log file (primarily used by `ims_master` and the LMTP server).

16.89.6 `maxlogfiles` Option

The `maxlogfiles logfile` option, *`component`*`.logfile.maxlogfiles`, specifies the maximum number of log files to retain.

16.89.6.1 Use with mmp Under logfile

For the MMP, the `maxlogfiles logfile` option, `mmp.logfile.maxlogfiles`, was new for Messaging Server 7u1.

16.89.7 `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).

16.89.7.1 Use with mmp Under logfile

For the MMP, the `maxlogfilesize logfile` option, `mmp.logfile.maxlogfilesize`, was new for Messaging Server 7u1.

16.89.8 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).

16.89.8.1 Use with mmp Under logfile

For the MMP, the maxlogsize logfile option, mmp.logfile.maxlogsize, was new for Messaging Server 7u1.

16.89.9 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).

16.89.9.1 Use with mmp Under logfile

For the MMP, the rollovertime logfile option, mmp.logfile.rollovertime, was new for Messaging Server 7u1.

16.89.10 rolloverpolicy Option

The rolloverpolicy logfile option, *component*.logfile.rolloverpolicy, specifies the policy of rolling over an active log file.

Starting with Messaging Server 8.0.2, this option is deleted. Rollover based on size is always performed, and rollover based on time is performed if the rollover manager process is enabled. DELETED: simplify rollover model.

- 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.

16.89.11 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.

16.89.11.1 Use with mmp Under logfile

For the MMP, the syslogfacility logfile option, mmp.logfile.syslogfacility, was new for Messaging Server 7u1.

16.90 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.

Autorestart of Messaging Server components is triggered if/when `msprobe` detects a "problem" with a component of Messaging Server, at which point `msprobe` requests that the Watcher attempt a restart of that "troubled" component of Messaging Server.

16.90.1 `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, MSHTTP, Job Controller, Dispatcher, and MMP servers.

16.90.2 `timeout` Option Under autorestart

The `timeout` Autorestart option, (`autorestart.timeout` in Unified Configuration, or `local.autorestart.timeout` in legacy configuration), specifies a failure retry time-out, in seconds. The default is `600` seconds. 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 `schedule.task:msprobe.crontab` option setting for msprobe's schedule).

16.91 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`.

For debugging of certificate map operations, see also the `certmap` keyword of the `debugkeys` option.

16.91.1 `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.

16.91.2 `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.

16.91.3 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.

16.91.4 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.

16.92 Base domainmap options

The only domainmap option is debug.

See also the usedomainmap Auth option.

The imsimta test -domain_map utility may be used to verify domain definitions.

16.92.1 debug Option Under domainmap

The base.domainmap.debug option sets the level of debugging messages for the domain map library code.

Chapter 17 Scheduler options

The Messaging Server Scheduler schedules and initiates execution of various periodic jobs for Messaging Server. These jobs may include the following named `task`s:

1. `return_job`, the MTA's message return job (message bouncer job), that returns (bounces) excessively old, undelivered messages,

2. `expire`, the Message Store's message expiration job, that deletes *from disk* messages that users have deleted from their mailboxes,

3. `msprobe`, checking whether Messaging Server components such as server processes are available ("up") and responsive,

4. `purge`, the MTA's log file purge job, that purges "older" MTA log files,

5. `snapshot`, the Message Store's database "snapshot" job, that captures a current-moment "snapshot" of the messages in the Message Store,

6. `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.

The Scheduler does not have its own `logfile` option group. This means that in unified configuration Scheduler debugging is controlled by the `base.logfile` option group. For example, Scheduler debug output to the default log can be enabled by setting:

```
msconfig> set base.logfile.loglevel debug
```

Note that this enables debug output for other utilities, e.g., msprobe, as well.

17.1 `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.

17.2 enablelog Option

To enable output from the Scheduler's tasks to go to separate log files in the *DATAROOT*/log directory, set enablelog to 1. This creates logfiles named <task-name>.log.<unix-timestamp> each time a scheduler task is executed. These log files are not removed automatically, so customers using this option will need to configure their own mechanism to remove older log files of this format.

This option is not refreshable; the scheduler must be stopped and restarted if this option is changed.

17.3 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:

1. 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,

2. msprobe, checking whether Messaging Server components such as server processes are available ("up") and responsive,

3. purge, the MTA's log file purge job, that purges older MTA log files,

4. return_job, the MTA's message return job (message bouncer job), that returns (bounces) excessively old, undelivered messages,

5. snapshot, the Message Store's database "snapshot" job, that captures a current-moment "snapshot" of the messages in the Message Store,

6. snapshotverify, the Message Store's database "snapshot verification" job, that verifies whether snapshots are sound.

Individual Scheduler tasks can themselves be enabled or disabled through the use of the enable task option. For example, the following commands disable msprobe and explicitly enable imexpire:

```
msconfig> set schedule.task:msprobe.enable 0
msconfig# set schedule.task:expire.enable 1
```

Almost all Scheduler settings are refreshable: New tasks can be added, task parameters can be changed, disabled tasks can be enabled, and as of Messaging Server 8.0.1.2, enabled tasks can be enabled. In all of these cases a `refresh imsched` will cause the changes to take effect.

At present only the outright deletion of a task requires a Scheduler restart:

```
stop-msg sched
start-msg sched
```

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.

17.3.1 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.

17.3.2 expire task options

The `expire` Scheduler task has a few options: `enable` to enable automatic scheduling of execution of `imexpire` and `crontab` to control that schedule.

Note that configuration of what `imexpire` actually does when it runs involves additional other, potentially complex, configuration. See Message Store expire options, Message Store expirerule options, the `expiresieve` Message Store option, and the discussion of `imexpire` invoking spamfilter packages for discussions of configuration of what `imexpire` actually does when it executes.

17.3.2.1 enable Option Use With expire Under task

The `enable` Scheduler `task` option for the `expire` task controls whether the expire task should be scheduled. It defaults to the setting of the `store.enable` option (Unified Configuration) or `local.store.enable configutil` parameter (legacy configuration).

17.3.2.2 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
```

17.3.3 msprobe task options

The `msprobe` Scheduler task, which runs `msprobe` to check on the status of various services, has a couple of options.

Note the distinction between these options set under the Scheduler task `msprobe`, `schedule.task:msprobe.`*`option-name`*, which control the *scheduling of execution* of `msprobe`, compared to the separate set of msprobe options controlling the actual *operation* of `msprobe`, set as `msprobe.`*`option-name`* or under specific service probes as `msprobe.probe:`*`specific-service-name.option-name`*.

17.3.3.1 `enable` Option Use With msprobe Under task

The `enable` Scheduler `task` option for the `msprobe` task controls whether the msprobe task should be scheduled.

17.3.3.2 `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
```

17.3.4 purge task options

The `purge` Scheduler task, which purges "old" versions of MTA log files, has a couple of options.

(Note the distinction between this MTA log file purge job, *vs*. the similarly named `impurge` job/daemon which purges old Message Store messages.)

17.3.4.1 `enable` Option Use With purge Under task

The `enable` Scheduler `task` option for the `purge` task controls whether the MTA's log file purge 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, this instead defaults to 1 if any of the following are set:

dispatcher.enable option (Unified Configuration), the local.dispatcher.enable configutil parameter (legacy configuration), the job_controller.enable option (Unified Configuration), the local.job_controller.enable configutil parameter (legacy configuration); and otherwise will default to 0. As the new dispatcher and job_controller enable options default to the mta.enable setting, upgrading customers should see no behavior change.

17.3.4.2 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
```

17.3.5 return_job options

The return_job Scheduler task has a couple of options. The return_job checks for messages in the MTA queue area that have not yet been delivered, and then generates delay warning messages, or bounces the messages, as appropriate and configured via the *notices channel options and return_units MTA option.

A number of MTA options modify operation of the return_job. In particular the return_split_period and return_cleanup_period MTA options affect the return_job's management of MTA transaction log files, while for debugging the return_job, see the return_debug and return_verify MTA options.

17.3.5.1 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.

17.3.5.2 crontab Option Use With return_job Under task

The crontab Scheduler task option for the return_job task controls the interval for running the MTA 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.

Initial configuration sets this to:

```
msconfig> show schedule.task:return_job.crontab
role.schedule.task:return_job.crontab = 30 0 * * * lib/return_job
```

17.3.6 snapshot task options

See also the Scheduler's `snapshotverify` task options. See also the `snapshotdirs` and `snapshotpath` Message Store options.

17.3.6.1 `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).

17.3.6.2 `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
```

17.3.7 snapshotverify task options

See also the Scheduler `snapshot` task options.

17.3.7.1 `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.

17.3.7.2 `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 18 Watcher options

The Watcher has just a few options.

See also `msprobe` options and Base autorestart options.

18.1 `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.

18.2 `port` Option Under watcher

The `port` Watcher option specifies the `watcher` listen port. The default is `49994`.

18.3 `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 19 msprobe options

There are a few options affecting msprobe operation. msprobe's service-specific probes can also individually override some of the general msprobe values.

Note the distinction between these options controlling *operation* of `msprobe`, set as `msprobe.option-name`, compared to the separate set of `msprobe` Scheduler task options which control the *timing of execution* of `msprobe`, and which are set as `schedule.task:msprobe.option-name`.

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 Base autorestart options especially `base.autorestart.enable`) and Watcher options, and Alarm options, respectively.

See also the `stressperiod` and `stressfdwait` base options (which currently affect only the MMP).

19.1 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.

19.2 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.

19.3 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).

19.4 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`
- `lmtp`
- `metermaid`
- `pop`
- `smtp`
- `submit`

So for instance:

```
msconfig> set msprobe.probe:submit.timeout 120
```

Chapter 20 Alarm options

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.

20.1 `noticehost` Option

The `noticehost` alarm option specifies the SMTP host to which msprobe should submit warning messages. 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. Do not use the SMTP server being monitored (the local instance) as the `noticehost`. To disable alarm message, set `noticeport` to 0.

20.2 `noticeport` Option

The `noticeport` alarm option specifies the SMTP port to which msprobe will connect when submitting alarm messages. A value of 0 will disable alarm message submission.

20.3 `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.

20.4 `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.

20.5 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 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).

20.6 smtpauthpassword Option Under alarm

The smtpauthpassword alarm option specifies the password that will be used when msprobe submits mail to the MTA. See also smtpauthuser.

20.7 smtpauthuser Option Under alarm

When msprobe submits, SMTP authentication will be used if both smtpauthuser and smtpauthpassword are set. These two Alarm options specify the administrative user name and password that are used to submit alarm.

20.8 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 serverreponse.

20.8.1 description Option Use With diskavail Under system

The alarm.system:diskavail.description option specifies the description for the diskavail alarm.

20.8.2 description Option Use With serverresponse Under system

The alarm.system:serverresponse.description option specifies the description for the serverresponse alarm.

20.8.3 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.

20.8.3.1 Use with diskavail Under system

The alarm.system:diskavail.statinterval option specifies the interval in seconds between disk availability checks. The default is 3600 seconds. Set to 0 to disable checks of disk usage.

20.8.3.2 Use with serverresponse Under system

The alarm.system:serverresponse.statinterval option specifies the interval in seconds between checks on server responsiveness. The default is 600 seconds. Set to 0 to disable checking of server response.

20.8.4 threshold Option

The threshold option, available under the alarm system named groups diskavail and serverresponse, specifies the threshold measurement for triggering an alarm.

20.8.4.1 Use with diskavail Under system

The alarm.system:diskavail.threshold option specifies the percentage of disk space availability below which an alarm is sent. The default is 10 percent.

20.8.4.2 Use with serverresponse Under system

The alarm.system:serverresponse.threshold option specifies the server response time, in seconds, triggering an alarm. The default is 10 seconds.

20.8.5 thresholddirection Option

Specifies whether an alarm is issued when the measurement is greater than (1) or less than (-1) the specified threshold.

20.8.5.1 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). The default is -1.

20.8.5.2 Use with serverresponse Under system

The alarm.system:serverresponse.theshlddirection option specifies whether an alarm is issued when server response time is greater than (1) or less than (-1) the threshold. The default is 1.

20.8.6 warninginterval Option

The warninginterval Alarm system option specifies, for a named system, the interval in hours between subsequent issuances of the alarm.

20.8.6.1 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. The default is 24 hours.

20.8.6.2 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. The default is 24 hours.

20.9 smtptls Option Under alarm

The smtptls Alarm options specifies whether to use TLS for SMTP connections; that is, whether alarm uses the SMTP extension STARTTLS and negotiates TLS use.

Chapter 21 Auth options

A number of options may be set under the auth group to affect authentication in general.

For debugging of authentication, see also the debugkeys option. For the format for addresses used in authentication, see also the ldap_domain_attr_uid_separator and loginseparator Base options and defaultdomain IMAP, POP, and MMP *et al.* option.

Caching of authentication results is controlled by the authcachesize and authcachettl Base options.

Regarding client certificate based authentication, see also the Base certmap options.

For configuration of authentication use for SMTP message submission, see TLS and SASL channel options and Password and TLS MTA options and the AUTH_ACCESS mapping table.

For tracking and penalization of "bad guy" failed authentication attempts, see the bg* options such as bgpenalty, settable at Base, IMAP, POP, and IMAP Proxy and POP Proxy levels. Or for monitoring or penalizing failed SMTP AUTH attempts, see various example uses of the LOG_ACTION mapping table.

See also the allowanonymouslogin option available under imap, pop, and http.

21.1 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.

21.2 authenticationserver Option

The authenticationserver Auth option specifies the hostname and port for a third-party authentication service to use for authentication. This option is also available at imapproxy and popproxy level (to override, for the respective server, the general Auth option). 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.

21.3 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.

21.4 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 incompliant 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

Adminstrator 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.

21.5 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).

For Messaging Server 8.0.2, the default was changed to use %P as the attribute name instead of 'uid'. The %P substitution refers to the LDAP attribute name specified by the ldap_permid option. This means it is only necessary to change the ldap_permid option to control the canonical user identity.

21.6 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.

21.7 requireauthenticationserver Option

The requireauthenticationserver option is available under auth and under imapproxy and popproxy.

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.

21.8 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).

21.9 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 the searchfordomain Auth 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's value) 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.

21.10 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.

Note that the `imsimta test -domain_map` utility can perform certain basic checks of domain layout validity.

Chapter 22 sectoken options

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.

22.1 `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 23 Deployment Map options

Several options relate to the Deployment Map service.

See also the `properties` Base option.

Note that msprobe can probe for whether the Deployment Map server is running; see msprobe's probe options.

23.1 `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.

23.2 `capability_starttls` Option Under deploymap

The `capability_starttls` deploymap option, when set to `1` (the default), causes the Deployment Map server to enable the STARTTLS extension. Advertising TLS (SSL) support does not mandate its use. Requiring use of TLS (SSL) is accomplished with the `deploymap.sslusessl` option.

23.3 `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.

23.4 `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.

23.5 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. The default is 4570.

23.6 properties Option

The properties base option and rolename option are used only during initial configuration by a Deployment Map client to convey information to a remote Deployment Map server.

23.7 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 Deployment Map option (local.deploymap.serverhost for legacy config). That option specifies the hostname or IP address of the host running the Deployment Map server.

23.8 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 Deployment Map server listens on.

23.9 sslusessl Option Under deploymap

Setting sslusessl to 1 instructs Deployment Map clients and servers to require the use of SSL (also known as TLS). When enabled on the server, the server will not allow clients to authenticate without first having successfully negotiated SSL. When enabled on the client, the client will negotiate SSL before attempting to authenticate. If the server does not advertise the TLS capability, then the client will not authenticate.

A server with this option enabled must also have the deploymap.capability_starttls option enabled (local.deploymap.capability.starttls in a legacy configuration).

23.10 userid and passwd Options

The userid and passwd Deployment Map options (local.deploymap.userid and local.deploymap.passwd for legacy config) are the shared secret pair used by Deployment Map clients to authenticate with the Deployment Map server. They will be sent in the clear unless the use of SSL has been enabled with the deploymap.sslusessl Deployment Map option (local.deploymap.sslusessl in legacy configuration).

A password and userid pair must be used in order for a client to connect to the Deployment Map server and be recognized by the server as being online. Without the authentication pair,

a client may only request the current deployment map file and list of online hosts. This shared secret pair must also be used to add or remove a host from the deployment map file.

Chapter 24 rollovermanager options

The Messaging Server's `rollovermanager` process performs time-based roll over for nslog-style log files. Criteria for log file "roll over" for these components is configured via the usual `logfile` options for the respective service.

The `rollovermanager` log file is controlled by `rollovermanager.logfile.*` options and the process is controlled by `rollovermanager.enable`.

For 8.0, the rollover manager only performed time-based rollover for three log files (imap, pop, transactlog). This was expanded to cover all log files with the 8.0.2 release.

24.1 `enable` Option Under rollovermanager

The `enable` rollovermanager option enables its operation.

Chapter 25 Messaging Server Ports

The Messaging Server uses different TCP (and UDP) Ports for different services. Some of these are intended for use by end-users, some for use by the deployment, and some for use only by the local host. This table lists the ports used by the product and their typical values. It's important to appropriately configure firewalls to only expose the intended ports.

Table 25.1 Table of TCP/IP Ports used by Messaging Server

Service	Port	Usage	Option	Description
Cassandra client	9042	product deployment	`store.casconnectpoints`	Cassandra Query Protocol used by Cassandra Message Store.
Certificate Port	55443	product deployment	`http.cert_port`	Private certificate validation protocol for Convergence S/MIME client.
ClamAV Port	3310	product deployment	PORT ClamAV plugin option	Used to talk to ClamAV via libclamav.so spam filter plugin.
dbreplicate server & client	55000	product deployment	`store.dbreplicate.port`	Berkeley DB replication for classic store automatic failover.
DNS client	53 (UDP & TCP)	nameserver in `/etc/resolv.conf`	N/A	DNS client used for MTA MX lookups, DNS RBL lookups, etc.
Elasticsearch client	9200	product deployment	`elasticsearch.port`	Port used to communicate with Elasticsearch.
ens	7997	product deployment	`ens.port`	Non-SSL product internal event notification service. Client port is specified via the `ensport` notifytarget option with `ensusessl` determining if SSL is used.
ens ssl	8997	product deployment	`ens.sslport`	SSL product internal event notification service.
icap client	1344	product deployment	PORT ICAP plugin option	Typically used to talk to Nortan AV via libicap.so spam filter plugin; RFC 3507.
imap	143	end-user (or product deployment with MMP)	`imap.port`	Non-SSL Internet Message Access Protocol (IMAP) RFC 3501.
imaps	993	end-user (or product deployment with MMP)	`imap.sslport`	SSL Internet Message Access Protocol (IMAP) RFC 8314, RFC 3501.
isc client	8070	product deployment	`isc_client.server_port`	Indexed search converter service used to support Elastic Search and Cassandra store search. Use `isc_client.sslusessl` to control SSL
isc server	8070	product deployment	`isc.server_port`	Indexed search converter service used to support Elastic Search and Cassandra store search. Use `isc.sslusessl` to control SSL
job controller	27442	host internal	`job_controller.tcp_ports`	Internal service used to notify job controller of message enqueues.
LDAP client	389 or 636	product deployment	`base.ugldapport`	Lightweight Directory Access Protocol (LDAPv3) RFC 4511 used for domain, user, and group lookups and authentication. Use 636 for SSL.
LMTP client	225	product deployment	`channel:tcp_lmtpcs.port`	Local Mail Transfer Protocol product interface based on RFC 2033.
LMTP server	225	product deployment	`dispatcher.service:LMTPSS.tcp_ports`	Local Mail Transfer Protocol product interface based on RFC 2033.
ManageSieve server	4190	end-user	`dispatcher.service:MANAGESIEVE.tcp_ports`	Protocol for managing Sieve scripts in RFC 5804.
memcache client	11211	product deployment	`mta.memcache_port`	A de-facto standard protocol used to talk to a fast key/value database typically used for deployment-wide tracking, metering, and rate-limiting.
metermaid client	63837	product deployment	`metermaid_client.server_port`	Product private RAM database similar to memcached. Use of memcached/Redis is recommended instead of metermaid for newer deployments.
metermaid server	63837	product deployment	`metermaid.port`	Product private RAM database similar to memcached. Use of memcached/Redis is recommended instead of metermaid for newer deployments.
mmp-imap	143	end-user	`imapproxy.tcp_listen:imapproxy1.tcp_ports`	Non-SSL MMP IMAP proxy RFC 3501.
mmp-imaps	995	end-user	`imapproxy.tcp_listen:imapproxy1.ssl_ports`	SSL MMP IMAP proxy RFC 8314, RFC 3501.

mmp-pop3	110	end-user	`popproxy.tcp_listen:popproxy1.tcp_ports`	Non-SSL MMP POP3 proxy RFC 1939.
mmp-pop3s	993	end-user	`popproxy.tcp_listen:popproxy1.ssl_ports`	SSL MMP POP3 proxy RFC 8314, RFC 1939.
mtqp client	1038	product deployment or intranet deployments	`mta.mtqp_port`	Message Tracking Query Protocol (RFC 3887) with private extensions for message recall.
mtqp server	1038	product deployment or external tracking/recall clients	`dispatcher.service:MTQP.tcp_ports`	Message Tracking Query Protocol (RFC 3887) with private extensions for message recall.
pop3	110	end-user (or product deployment with MMP)	`pop.port`	Non-SSL Post Office Protocol version 3 (POP3) RFC 1939.
pop3s	993	end-user (or product deployment with MMP)	`pop.sslport`	SSL Post Office Protocol version 3 (POP3) RFC 8314, RFC 1939.
Redis client	6379	product deployment	`redis.port`	Protocol used to talk to a Redis server key/value database typically used for deployment-wide tracking, metering, and rate-limiting.
Redis sentinel client	26379	product deployment	TBD	Protocol used to talk to a Redis sentinel server to determine the active Redis master host for a named Redis master/slave pool.
smpp client	none	product deployment	`sms_gateway.smpp_relay.server_port`	Relay port for SMS gateway via Short Message Peer-to-Peer (SMPP) service.
snmp server	161 (UDP)	product deployment	`snmp.port`	SNMP monitoring port (RFC 2789, RFC 3411-3418).
smtp server	25	Internet MTAs	`dispatcher.service:SMTP.tcp_ports`	Simple Mail Transfer Protocol Relay Service (SMTP relay) RFC 5321.
Socks client	1080	product deployment	`socksport` channel option	Used to traverse a SOCKS 5 firewall (RFC 1928, RFC 1929).
Solr client	8983	product deployment	`store.solrconnectpoints`	Used by Cassandra store to bootstrap Datastax Solr search indexing. No longer used starting with Messaging Server 8.1.
Spamd client	783	product deployment	PORT Spam Assassin plugin option	Used to talk to Spam Assassin via libspamass.so spam filter plugin.
submission client	25	product deployment	`http.smtpport`, `alarm.noticeport`	Webmail & alarm SMTP Message Submission RFC 6409.
submission server	587	end-user	`dispatcher.service:SMTP_SUBMIT.tcp_ports`	Non-SSL Message Submission (SMTP submission) RFC 6409.
submissions server	465	end-user	`dispatcher.service:SMTP_SUBMIT.ssl_ports`	SSL Message Submission (SMTP submission) RFC 8314, RFC 6409.
watcher	49994	host internal	`watcher.port`	Private service to detect stressed and crashed processes for self-repair.
wmap / mshttpd	8990	product deployment	`http.port`	Non-SSL private webmail proxy layer between Convergence and IMAP, based on HTTP.
wmaps / mshttpd	8991	product deployment	`http.sslport`	SSL private webmail proxy layer between Convergence and IMAP, based on HTTP.

Part IV 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.

For configuration of some automatic maintenance jobs relating to the Message Store, see Scheduler task options.

Chapter 26 Message Store options

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
```

For example:

```
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

26.1 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.

26.2 admins Option

The admins Message Store option takes a space separated list of user ids with message store administrator privileges. The default is the user id admin. If single-valued or not set, the MMP will use this as the default value for its storeadmin option.

26.3 allowbadmailbox Option

The allowbadmailbox Message Store restricted option will disable certain mailbox validity checks in the message store. A value of 1 will disable Unicode normalization and a value of 2 will disable validity checks of the IMAP international naming convention. Use of this can have unpredictable results, including breaking ISS synchronization, causing backup problems and breaking IMAP clients. It exists primarily for testing purposes and as a contingency in case there are unexpected problems with ICU normalization. This option may be removed in a future release.

This option is ignored if Cassandra store is used or if an IMAP client negotiates UTF8=ACCEPT -- in both cases bad mailbox names are unconditionally forbidden.

Customers using ISS should never set this option on a production system; ISS requires all mailbox names to be RFC 5198 compliant.

26.4 autounsubscribe Option

Cassandra store only. The autounsubscribe Message Store option controls whether folders are unsubscribed by imapd automatically when they are deleted. The default is 1, meaning that folders are unsubscribed automatically. This option has no effect in classic store. Classic store does not unsubscribe folders automatically.

26.5 autorepair Option

The autorepair Message Store option may be set to repair damaged mailboxes automatically.

26.6 autorepairdebug Option

The autorepairdebug Message Store option enables the backup of mailbox index files, before repair, to the /storedebug subdirectory under the Messaging Server temporary file directory specified by base.tmpdir in Unified Configuration (local.tmpdir in legacy configuration). The maximum number of backup mailboxes is 10.

Note: Sites should remove the files and directories under /storedebug manually when they are not needed.

26.7 backupdir Option

The backupdir Message Store option specifies the directory for backup image of Message Store data.

26.8 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.

See also the backup_group options.

26.9 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.

26.10 cachesynclevel Option

The cachesynclevel Message Store option controls the synchronization level for the 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.

26.11 casconnectpoints Option

The casconnectpoints Message Store option specifies a space-separated list of Cassandra Cluster connect points (ip addresses). If not specified, this option defaults to the IPv4 loopback address (127.0.0.1). specified, this option defaults to the local host (127.0.0.1).

26.12 solrconnectpoints Option

The solrconnectpoints Message Store option specifies a space-separated list of Cassandra Cluster connect points (ip addresses) for IMAP solr search. If not specified, casconnectpoints is used.

26.13 msgconnectpoints Option

The msgconnectpoints Message Store option specifies a space-separated list of Cassandra Cluster connect points (ip addresses) for msg keyspace access. If not specified, casconnectpoints is used.

26.14 cacheconnectpoints Option

The cacheconnectpoints Message Store option specifies a space-separated list of Cassandra Cluster connect points (ip addresses) for cache keyspace access. If not specified, casconnectpoints is used.

26.15 casusername Option

The casusername Message Store option specifies the username for cassandra cluster authentication.

26.16 caspassword Option

The caspassword Message Store option specifies the password for cassandra cluster authentication.

26.17 casmetarf Option

The casmetarf Message Store option specifies the replication factor for the store meta keyspace. If not specified, this option defaults to 1.

26.18 casmsgrf Option

The casmsgrf Message Store option specifies the teplication factor for the store msg keyspace. If not specified, this option defaults to 1.

26.19 cassolrrf Option

The cassolrrf Message Store option specifies the replication factor for the store solr keyspace. If not specified, this option defaults to 1.

26.20 cascacherf Option

The cascacherf Message Store option specifies the replication factor for the store cache keyspace. If not specified, this option defaults to 1.

26.21 casmaxconnectionsperhost Option

RESTRICTED: The casmaxconnectionsperhost Message Store option specifies the maximum number of connections per host. If not specified, this option defaults to 500.

26.22 casnumthreadsio Option

RESTRICTED: The casnumthreadsio Message Store option specifies the number of IO threads that will handle Cassandra query requests. If not specified, this option defaults to 5.

26.23 cascasopretrycount Option

RESTRICTED: The cascasopretrycount Message Store option specifies the cassandra CAS operation retry count upon write timeout. If not specified, this option defaults to 3.

26.24 cascasopretryintervalinms Option

RESTRICTED: The `cascasopretryintervalinms` Message Store option specifies the cassandra CAS operation retry interval (in milliseconds) upon write timeout. If not specified, this option defaults to `10`.

26.25 caskeyspaceprefix Option

The `caskeyspaceprefix` Message Store option specifies a prefix to all Cassandra keyspace names used by the Message Store. This can be used to have multiple message stores in the same Cassandra ring. This should not be changed on an operational server. This option defaults to "ms_".

26.26 cascachedc Option

The `cascacheadc` Message Store option specifies the datacenter of the cache keyspace. This option defaults to "`DC1`".

26.27 casmetadc Option

The `casmetadc` Message Store option specifies the datacenter of the meta keyspace. This option defaults to "DC1".

26.28 casmsgdc Option

The `casmsgdc` Message Store option specifies the datacenter of the msg keyspace. This option defaults to "DC1".

26.29 cassolrdc Option

The `cassolrdc` Message Store option specifies the datacenter of the solr keyspace. This option defaults to "`DC1`".

26.30 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.

26.31 checkmailhost Option

The `checkmailhost` Message Store option enables checking that the user `mailhost` attribute matches this server.

26.32 cleanupage Option

The cleanupage Message Store option specifies the age (in hours) of an expired or expunged message before purge will permanently remove it. The minimum allowed value is 1; the maximum is 2400 (100 days).

26.33 cleanupsize Option

The cleanupsize Message Store option specifies the minimum number of expunged messages before purge will permanently remove them.

26.34 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 enabled for the Message Store).

Note that there is also a separate dbcachesize option available at the Message Store msghash level, store.msghash.dbcachesize.

26.35 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 * number of message store processes) + (100 * 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).

26.36 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.

26.37 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.

26.38 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.

26.39 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 mboxlist 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.

26.40 dbtype Option

The dbtype Message Store option specifies the Message Store type. Use 'bdb' for the classic message store based on files and Oracle Berkeley DB or 'cassandra' for the Cassandra message store. The Cassandra Message Store was introduced in Messaging Server 8.0.2 and presently requires Unified Config. See the installation guide for additional requirements to run Cassandra Mesage Store.

26.41 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.

26.42 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.

26.43 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.

26.44 defaultpartition Option

The defaultpartition Message Store option specifies the default partition. The default is "primary". Only applicable on INBOX. Subfolders will be created in the partition of the parent folder.

For further configuration of partitions, see the Partition options.

26.45 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.

26.46 encryptnew Option

RESTRICTED: The encryptnew Message Store option enables encrypting new messages.

26.47 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.

26.48 expiresieve Option

The expiresieve Message Store option enables use of Sieve scripts in store.expirerule file rule sets; it particular, it enables application of sieve attributes in such files.

26.49 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.

26.50 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.

26.51 folderlockcount Option

The folderlockcount Message Store option sets the maximum number of folder locks. The minimum allowed value is 1000; the maximum is 100000.

26.52 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.

26.53 indexmapreadonly Option

The indexmapreadonly Message Store option is a potential optimization for NFS on some operating systems. Normally Messaging Server opens the index file for read/write access and maps that file for read-only use. When this option is set, Messaging Server will open the index file twice, once in read/write and once in read-only mode and memory map the read-only file descriptor. This may reduce performance on some filesystems (due to the extra file open operation), but may be helpful if you're seeing high system CPU usage in munmap on NFS.

26.54 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.

26.55 keypass Option

RESTRICTED: Keystore password.

26.56 keylabel Option

RESTRICTED: The keylabel Message Store option specifies the label of the Message Store key in the keystore.

26.57 listimplicit Option

When the listimplicit Message Store option is set to "1", implicitly shared folders will appear in lists performed by store admins.

26.58 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. Starting with Messaging

Server 8.0.1, expunge details are logged to the transactlog if it is enabled. Starting with Messaging Server 8.0.2, when the transactlog is enabled, this option is ignored and the actions option controls whether expunge events are logged.

26.59 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.

26.60 maxcachefilesize Option

The maxcachefilesize 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.

26.61 maxfolders Option

The maxfolders Message Store option specifies a maximum number of folders per user. Set to 0 (the default) for infinite.

26.62 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.

26.63 maxmessages Option

The maxmessages Message Store option specifies a maximum number of messages per folder.

26.64 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.

26.65 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.

Note that enbling the overquotastatus Message Store option causes the quotaoverdraft Message Store option to be enabled automatically.

26.66 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.

26.67 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%mailbox3*", where *mailbox1*, *mailbox2* and *mailbox3* 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
```

26.68 quotaenforcement Option

The quotaenforcement Message Store option enables quota enforcement. When off, the quota database is still updated, but messages are always delivered.

26.69 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 the warning threshold. In default operation (quotaoverdraft not enabled), the warning threshold is determined by the quotawarn Message Store option (legacy configuration, store.quotawarn). But when quotaoverdraft has been enabled, either explicitly or implicitly via overquotastatus, then instead the warning is generated only when a user actually exceeds their quota.

There is support for the following variable substitutions:

[ID]	userid
[DISKUSAGE]	disk usage
[DOMAIN]	user's domain (new in 8.0.1.3)
[NUMMSG]	number of messages
[PERCENT]	`store.quotawarn`percentage
[POSTMASTER]	postmaster address for user's domain (new in 8.0.1.3)
[QUOTA]	mailquota attribute value
[MSGQUOTA]	mailmsgquota attribute value

The message must contain a header (with at least a subject line), followed by `$$`, then the message body. The `$` represents a new line.

Note that the From: header line on the notification message will be:

```
From: Mail Administrator <Postmaster@base.hostname>
```

substituting in the value of `base.hostname` (or `local.hostname` in legacy configuration).

26.70 quotaexceededmsginterval Option

The `quotaexceededmsginterval` Message Store option specifies the interval (in days) to wait before sending another quota exceeded message.

26.71 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.

26.72 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.

Note that the `quotanotification` option must be disabled (at least temporarily) when using the `imquotacheck` utility to generate quota warning messages, so that the quota database's last-warn-time can be properly updated.

26.73 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.

Note that enabling quotaoverdraft means that quota enforcement does not start until *after* a user goes over quota, and in particular, means that the quotawarn Message Store option (used to warn users that they are nearing their quota limit when strict, no-overdraft, quota limits are in effect) does not take effect.

26.74 quotawarn Option

The quotawarn Message Store option specifies the percentage of quota that must be exceeded before clients are sent an over quota warning.

Note that such quota warning messages are only generated automatically when strict quota enforcement is in effect: that is, when the quotaoverdraft Message Store option is *not* enabled (whether explicitly or implicitly due to overquotastatus having being enabled). To generate quota warning messages before a user has exceeded their quota when quotaoverdraft is permitted, instead see the imquotacheck utility.

26.75 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.

26.76 searchengine Option

The searchengine Message Store option specifies the Message search engine type. Use 'iss' for the ISS search engine on Classic store (bdb); 'dse' for the DSE/Solr search engine on DSE Cassandra message store; Or, 'elastic' for the Elasticsearch engine on Classic message store and Cassandra store. The DSE Cassandra Message Store with Solr search engine was introduced in Messaging Server 8.0.2 and is no longer supported starting with Messaging Server 8.1. The Classic Store with Elasticsearch engine was introduced in Messaging Server 8.0.2.2. The Cassandra Message Store with Elasticsearch engine was introduced in Messaging Server 8.0.2.3. Both Open Source Cassandra and DSE Cassandra are supported. See the Index and Search section for additional requirements to run Elasticsearch. When searchengine isn't set, the messaging Server uses DSE/Solr on Cassandra store, ISS if ISS is enabled or brute force on Classic store.

26.77 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.

26.78 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.

26.79 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>
```

26.80 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.

26.81 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.

26.82 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.

26.83 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.

26.84 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.

26.85 undeleteflag Option

When the undeleteflag Message Store boolean option is set, it causes imapd to set an explicit $undeleted message flag when the \Deleted flag is removed. This feature allows clients to detect when a previously deleted message is no longer deleted, which can be helpful when providing a gateway to a protocol that has to treat this scenario as if a new message arrived. The $undeleted message flag has no additional special semantics and may be set or cleared by clients if the IMAP ACL on the mailbox permits flag changes.

26.86 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.

26.87 Message Store archive options

There are a number of options relating to the Message Store's facility for performing compliance archiving. (Note that if configured to perform archiving, the Message Store generates archive message copies when users perform IMAP APPEND operations. Message Store archiving is enabled by setting store.archive.compliance or See also use of the archive value of the action attribute in imexpire's store.expirerule files to perform archiving, rather than expiration, of messages meeting expiration rule set criteria. 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.

New in MS 8.0.1, the Message Store no longer supports the operational archiving option. DELETED: operational.

For debugging of archiving operations, see also the archive keyword of the debugkeys option.

26.87.1 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/.

26.87.2 compliance Option

The compliance Message Store archive option enables compliance archive.

26.87.3 operational Option

The operational Message Store archive option enables operational archive. DELETED: Removed in MS 8.0.2.

26.87.4 source_channel Option

The store.archive.source_channel option specifies the name of the MTA 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.

26.87.5 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.

26.87.6 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.

26.87.7 reportdir Option

Archive confirm report directory.

26.87.8 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.

26.87.9 posteddatemode Option

The posteddatemode Message Store archive 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 Nth 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.

26.87.10 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.

26.87.11 retrieveport Option

The retrieveport Message Store archive option specifies the archive retrieve server port.

26.87.12 retrieveserver Option

Archive retrieve server.

26.87.13 retrievetimeout Option

The retrievetimeout Message Store archive option specifies the archive retrieve timeout in seconds.

26.87.14 path Option Under archive

The path Message Store archive option specifies the archive injection directory.

26.88 Message Store checkpoint options

There are a few options relating to the Message Store's facility for checkpointing.

26.88.1 stresslimit Option

The stresslimit Message Store checkpoint option specifies the maximum stored checkpoint duration in seconds. Throttling starts when checkpoint duration exceeds this limit.

26.88.2 debug Option Under checkpoint

The debug Message Store checkpoint option (*i.e.*, store.checkpoint.debug), if set, enables stored checkpoint debug.

26.89 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
```

26.89.1 enable Option Under dbreplicate

Tne enable Message Store dbreplicate option enables the mboxlist database replication feature.

26.89.2 port Option Under dbreplicate

The port Message Store dbreplicate option specifies the mailbox list database replication TCP port number. The default is 55000. This will be used to listen for incoming connections.

26.89.3 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.

26.89.4 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.

26.89.5 twosites Option

The twosites Message Store dbreplicate 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.

26.89.6 queuemax Option

The queuemax Message Store dbreplicate options specifies the replication manager incoming queue size limit.

26.89.7 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.

26.89.8 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.

26.90 Message Store deadlock options

Under the deadlock group are a few Message Store options relating to deadlocks. See also the Message Store option (directly under store, rather than under store.deadlock) deadlockaggressive.

26.90.1 autodetect Option

The autodetect Message Store deadlock option sets whether all or just one thread resolves deadlock.

26.90.2 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.

26.91 Message Store expire options

The exploglevel option 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:

- schedule.task:expire.crontab
- store.expiresieve
- store.expirerule.* options
- store.cleanupage
- store.mailboxpurgedelay

Also see the Scheduler options for the expire task:

- schedule.task:expire.enable
- schedule.task:expire.crontab

26.91.1 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.

26.92 Message Store expirerule options

There are two separate ways to configure the Message Store's rules for expiring (purging) old (or otherwise undesired) messages. Global expiration rules *can* be defined via Message Store expirerule options (or the corresponding configutil parameters in legacy configuration), as discussed below; however, the preferred (more flexible, and more performant) approach is to define expiration rules using store.expirerule files. See the discussion of such files, as they allow more flexible, targetted rule sets (such as different per-user or per-partition rules), and they support additional criteria, and their use is more efficient for larger numbers of rules.

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, and the discussion of imexpire invoking spamfilter packages. For the scheduling of automatic execution of imexpire, see the Scheduler expire task.

26.92.1 deleted Option

Syntax: "and"|"or".

When the deleted Message Store expirerule option is set to "or", expire messages if the '\Deleted' IMAP system flag is set *or* other criteria in the expire rule are met. When set to "and", expire messages if the '\Deleted' IMAP system flag is set *and* other criteria in the expire rule are met.

26.92.2 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.

26.92.3 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.

26.92.4 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.

26.92.5 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.

26.92.6 messagedays Option

The messagedays Message Store expirerule option specifies the upper limit on the age of messages in the specified folder(s), using for message age the time when a message was originally deposited into the Message Store; messages "old" in terms of original delivery to the Message Store can thus be deleted even if such a message has only recently been moved to the current folder. Compare with the expire rule file setting "savedays", which uses the time since a message was placed into the current folder.

26.92.7 messagesize Option

The messagesize Message Store expirerule option specifies the size of an over-sized message.

26.92.8 messagesizedays Option

The messagesizedays Message Store expirerule option specifies the days an over-sized message should remain in a folder.

26.92.9 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.

26.93 Message Store folderquota options

Currently, the only Message Store folderquota option is enable.

26.93.1 enable Option Under folderquota

The enable Message Store folderquota option enables quota by folder.

26.94 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
```

26.94.1 enable Option Under messagetype

The `enable` Message Store `messagetype` option enables the Message Store's message typing feature.

26.94.2 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.

26.94.3 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.

26.94.4 contenttype Option

The `contenttype` Message Store `messagetypemtindex` 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 `messagetype mtindex` option specifies the Message Store name to use for such messages.

26.94.5 flagname Option

The `flagname` Message Store `messagetype 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.

26.94.6 quotaroot Option

The `quotaroot` Message Store `messagetype 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).

26.95 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
```

26.95.1 enable Option Under msghash

The `enable` Message Store `msghash` option enables message hash database.

26.95.2 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*1024*1024) will be used (other values will be silently adjusted into that range), with the default being 8388608 (8*1024*1024).

Note that this `store.msghash.dbcachesize` option is separate from the `store.dbcachesize` option, which controls a different cache size.

26.95.3 nummsgs Option

The `nummsgs` Message Store `msghash` option specifies the message hash database size.

26.96 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 expire task options

- schedule.task:expire.enable
- schedule.task:expire.crontab

26.96.1 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).

If store.enable is 0, the value of this option is ignored and the impurge daemon is not started. If store.enable is 1, the impurge daemon will run unless this option is explicitly set to 0.

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, using a Scheduler task. impurge cannot be executed manually by an administrator, nor 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
```

26.96.2 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.

26.96.3 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.

26.96.4 percentage Option

The percentage Message Store purge option specifies the percentage of expunged cache records (for a folder) before purge will permanently remove them.

26.96.5 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
```

26.97 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
```

26.97.1 `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.

26.97.2 `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.

26.97.3 `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.

26.97.4 `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).

26.98 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`

26.98.1 `restrictanyone` Option

The `restrictanyone` Message Store `privatesharedfolders` option disallows regular users sharing private folders to anyone.

26.98.2 `restrictdomain` Option

Classic store only. The `restrictdomain` Message Store `privatesharedfolders` option disallow regular users sharing private folders to users in another domain. This option has no effect in cassandra store. Regular users cannot share private folders to users in another domain in cassandra store.

26.98.3 `shareflags` Option

The `shareflags` Message Store `privatesharedfolders` option controls whether private shared folders share `\Seen` and `\Deleted` flags across users.

26.98.4 `user` Option Under publicsharedfolders

The `user` Message Store `publicsharedfolders` option specifies the public shared folder owner's store user identity. For non-ASCII users or domains, this should use the Net-Unicode (RFC 5198) form of the user and domain (IDN A-labels should not be used).

Chapter 27 message_language options

There are a few options available for localizing certain automatically generated messages. These options may be set under named `message_language` groups, *e.g.*, `message_language:`*`language-name.option-name`*.

27.1 `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 the warning threshold. In default operation (`quotaoverdraft` not enabled), the warning threshold is determined by the `quotawarn` Message Store option (legacy configuration, `store.quotawarn`). But when `quotaoverdraft` has been enabled, either explicitly or implicitly via `overquotastatus`, then instead the warning is generated only when a user actually exceeds their quota.

There is support for the following variables:

[ID]	userid
[DISKUSAGE]	disk usage
[DOMAIN]	user's domain (new in 8.0.1.3)
[NUMMSG]	number of messages
[PERCENT]	`store.quotawarn`percentage
[POSTMASTER]	postmaster address for user's domain (new in 8.0.1.3)
[QUOTA]	mailquota attribute value
[MSGQUOTA]	mailmsgquota attribute value

The message must contain a header (with at least a subject line), followed by $$, then the message body. The $ represents a new line.

27.2 `welcomemsg` Option Under message_language

The `welcomemsg` option specified under a named `message_language` specifies a localized welcome message for new Message Store users. The maximum size is 1 MB.

Syntax: "$" line separators, with headers.

Chapter 28 Partition options

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, such as `checkdiskusage` and `defaultpartition`.

28.1 `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 DATAROOT 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.

28.2 `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 (DATAROOT). If not specified, the value for the `partition:`*`partition-name`*`.path` option (in legacy configuration, the `store.partition.*.path` configutil parameter) is used.

28.3 `path` Option Under partition

The `path` partition option specifies the path name of a Message Store partition. If this is not a full path, this is relative to the `store/partition` subdirectory of the data directory (DATAROOT).

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.

Note that message expiration rules for this partition may be stored in files located in this directory; see `store.expirerule` files.

28.4 `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).

Note that the `defaultpartition` Message Store option defaults to "`primary`" -- so the definition of the partition named "`primary`" normally defines the partition that the Message Store uses by default.

Chapter 29 backup_group options

Under backup_group, there is a single option, re_pattern.

See also the backupexclude Message Store option.

29.1 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 30 Store Transaction Log Format

Message store XML transaction logging is enabled by setting the `messagetrace.activate` option to `transactlog`. Both the MTA and store have legacy transaction log formats that are deprecated in favor of the XML format. For a discussion of the MTA XML logging format, see the log_format MTA option.

Each log entry is a self-closing XML entity with a two-letter entity name that contains attributes (also with two-letter names). New attributes may be added and the attributes may be re-ordered in any patch release, so use of an XML-aware parser is recommended. The content of the log is intended to be extended in a backwards-compatible format (with possible exceptions for a major release), unlike the legacy store messagetrace and server log formats which are unstable. Due to the size of transaction log files on busy systems, use of a SAX-style parser is recommended.

To control what actions are logged, use the actions option (unified configuration only). To control what attributes are logged, use the actionattributes option. Note that the MTA uses a different mechanism to control what is logged; see the Transaction logging MTA options section.

30.1 XML Log Attributes Always Present in Store Transaction log

The following attributes appear on all Store Transaction Log entries and are thus not mentioned in the event-specific descriptions below:

- `pi` - Process id (integer). Note that the MTA uses a different format for this attribute documented in the log_process option.
- `sn` - service name (e.g., imap, pop, imquotacheck)
- `ts` - time stamp. Both MTA & store use ISO 8601 format as of 8.0.2; but in 8.0 the store used a legacy timestamp format.

30.2 XML Log Common Attributes

The following attributes may appear on several XML log entry event types with largely consistent meaning. When these are mentioned in the 'Common' attribute list for an event type, they will be included unless disabled by the actionattributes option.

- `ma` - IMAP Mailbox name (internal form)
- `mi` - Message id

- `om` - Source mailbox for copy/rename (`ma` is destination mailbox)
- `si` - session id (IMAP & POP): a unique integer identifier for a client session/connection.
- `tr` - transport information (MTA & store). Prior to 8.0.2, the store only included the client's address and port in this field.
- `us` - User name: the canonial authorization user identity (the permanent identity of the primary mail account being accessed). For more information on user identifiers see User Identifiers. Can also be [unauthenticated] when appropriate. For the `ac` action, the string "Admin" is used when this can't be determined (typically for mboxutil).

30.3 XML Log Entity Names and Specific Attributes:

`ac` - Access Control Change (IMAP only). Attributes include:

- Common: `ma`, `si`, `us`
- `ao` - Old ACL using permanent user identifiers. New in 8.0.2.
- `an` - New ACL using permanent user identifiers. New in 8.0.2.
- `nt` - Old and new ACL with ':' delimiter (Messaging Server prior to 8.0.2 only).

`bm` - Big Memory Allocation Event (new in MS 8.0.2.2). Attributes include:

- `nt` - Big memory function (malloc, calloc, realloc)
- `sz` - Bytes allocated
- `fn` - Source filename of allocation
- `ln` - Source line number of allocation

`cp` - Copy Message Event (new in MS 8.0.2.2). Attributes include:

- Common: `ma`, `om`, `si`
- `mc` - number of copied messages
- `sz` - total size of copied messages
- `su` - source IMAP UID set for copy operation
- `uc` - destination IMAP UID set for copy operation
- `uv` - IMAP UIDVALIDITY for destination
- `nt` - Error message on copy failure (omitted on success, new in MS 8.1.0.5.0)

`co` - Socket Connection (open/close). Attributes include:

- Common: `si`, `tr`
- `ac` - Action code. First letter is 'O' for connection open and 'C' for connection close. Subsequent letters are extensible flags. See MTA transaction log entry format for the

meaning of the subsequent flags for the MTA. Subcodes that can be used by the MMP and store include:

- `D` - Closed due to DNS RBL
- `I` - Closed due to internal/config error
- `L` - Closed due to connection limit
- `P` - Closed due to broken pipe
- `R` - Closed due to connection reset
- `S` - Closed due to socket error
- `T` - Closed due to timeout
- `W` - Closed due to TCP Access wrap filter
- `F` - Closed due to force kill, imsconnutil -k

- `at` - Store only: will be 'ssl' if SSL was used at connection open time or empty string if SSL was not used.
- `br` - bytes received during connection (new in MS 8.0.2).
- `bs` - bytes sent during connection (new in MS 8.0.2).
- `fu` - flag update page scan count (new in MS 8.0.2.2). A large number indicates potentially significant server CPU consumed by this user's client.
- `nm` - number of mailboxes selected during session (new in MS 8.0.2, imap/pop only).
- `nt` - In 8.0, contains unstructured information about the connection at connection close. Removed in MS 8.0.2 in favor of separate attributes.
- `rr` - Reason connection was rejected (new in 8.0.2.1, MMP only)
- `sb` - Search body count (new in MS 8.0.2.2). This counts the number of messages mapped for searching purposes by this user. This does not count searches performed by ISS, DSE or elastic search.
- `sd` - Session duration with HHH:MM:SS format (new in MS 8.0.2).
- `td` - Time spent on DNS RBL lookups in milliseconds (new in MS 8.0.2.1, MMP only).

`ex` - Expunge Action (store IMAP expunge/expire). Attributes include:

- Common: `ma`, `mi`, `si`
- `mc` - Messages in mailbox (post-expunge). Prior to MS 8.0.2 this attribute combined `me` with the pre-expunge message count using a '/' delimiter.
- `me` - Messages changed (for `ex` action, messages expunged). New in MS 8.0.2.
- `mi` - Message Id. Note that when this attribute is enabled, a separate expunge log entry is created for each message. If this attribute is not enabled, then only one expunge entry is created for each expunge operation.

- `no` - Node name (local host name or remote client IP & port).

`fc` - Flag Change Action (store +/-flags). New in MS 8.0.2.2.

- Common: `ma`, `si`
- `ac` - Action code; one of "S" for set, "C" for clear or "R" for replace.
- `fl` - Flag list (space delimited)
- `me` - Number of messages changed
- `sq` - IMAP modification sequence for this change
- `uc` - UIDs changed in IMAP uid set format
- `uv` - IMAP UIDVALIDITY for mailbox

`fe` - Fetch Message Action (POP & IMAP only). Attributes include:

- Common: `ma`, `mi`, `si`, `us`
- `fd` - Fetch decoding (b64, qp or omitted) (8.0.2 IMAP only)
- `fo` - Offset to message part in stored message (8.0.2 IMAP only)
- `fp` - Fetch offset into message part (8.0.2 IMAP only)
- `om` - Alternate for `ma` code (8.0.1 POP only)
- `sz` - Actual bytes fetched. For 8.0.2 this is a number. For earlier 8.0 versions, this instead contains a string combining: fetch start offset ":" fetch data size or "Binary:" followed by the offset into the message, the offset into the decoded data and the fetch data size (IMAP only).
- `ui` - IMAP UID for message (8.0.2 IMAP only)

`li` - Login/Authenticate Action (store/MMP). Attributes include:

- Common: `si`, `tr`, `us`
- `ae` - Integer authentication error code; see `nt` for description. Omitted if not known (MMP only, new in MS 8.0.2.1)
- `at` - Authentication Type (SASL mechanism name, ssl-port-cert, anonymous or plaintext)
- `bd` - badness delay (seconds) before next authentication attempt (MMP only, new in MS 8.0.2.1)
- `cs` - Ciphersuite used followed by TLS version. If SSL/TLS is not used, this will be 'noSSL'.
- `nt` - Authentication Error or Reply
- `ph` - Proxy host name from mailHost or affinity config (MMP only, new in MS 8.0.2.1).
- `pt` - Proxy transport information (MMP only, new in MS 8.0.2.1)
- `ua` - User authentication identity. This is the user whose password is used to authenticate; which differs from `us` when administrative proxy authentication is used (new in MS 8.0.2).

- `uo` - Original user identity. This is the identity originally specified by the client prior to canonicalization (MMP only, new in MS 8.0.2.1)

`lo` - Logout action; (POP-only, only if poplogmboxstat is set). Attributes include:

- Common: `si`, `tr`, `us`
- `ct` - Unix timestamp of POP login.
- `mc` - Number of messages not marked for deletion.
- `sz` - Total bytes in messages not marked for deletion.

`ma` - Message Append Action. Attributes include:

- Common: `ma`, `mi`, `si`, `us`
- `cx` - alternate name for session identifier (MS 8.0.1 only).
- `sz` - Total bytes in the appended message.
- `ui` - IMAP UID for message
- `uv` - IMAP UIDVALIDITY for message

`is` - ISC Convertion Action. Attributes include:

- Common: `us`
- `ma` - The user-folderId information .
- `ui` - IMAP UID for message
- `sz` - Total bytes in the message.
- `ez` - Total bytes sent to ElasticSearch
- `it` - Indexing of document in Elasticsearch response time in microseconds

`mc`, `md`, `mr` - Mailbox Create, Delete, Rename Actions (IMAP only). Attributes include:

- Common: `ma`, `om`, `si`, `us`
- `fi` - partition name (classic store only)
- `rt` - Mailbox Rename duration (introduced in 8.1.0.1)

`ms`, `mu` - Mailbox Subscribe, Unsubscribe Actions (IMAP only). Attributes include:

- Common: `ma`, `si`, `us`
- `fi` - namespace (IMAP2bis only)

`qc` - Quota Change (IMAP only). Attributes include:

- Common: `si`, `us`
- `ur` - Quota Root

- dq - Disk storage quota (number in KB)

qe - Quota Exceeded Action (quotacheck tool only). Attributes include:

- Common: us
- dq - Disk storage quota (number in KB)
- du - Disk storage usage (number in KB)
- mq - Message count quota (number)
- mc - Message count used (number)
- qt - Overquota Trigger (numeric percentage)
- qr - Quota Rule Name ('General' if not using a rule file)

se - Search, Sort or Thread Mailbox (IMAP only). Attributes include: (new in 8.0.2.2)

- Common: ma, si
- mc - Number of matching messages
- nm - Number of mailboxes searched (only counts local mailboxes when remote shared folders are present).
- nt - Error message on search failure (omitted on success)
- sb - Search body count. This counts the number of messages mapped for searching purposes for this search. This does not count searches performed by ISS, DSE or Elasticsearch.
- sf - Search flags (Q=Message sequence search, U=Uid search, I=iss, D=dse, E=elastic, C=classic, T=Thread, X=Context, S=Sort, M=multi-mailbox search)
- td - Time passed during search in milliseconds (omitted if 0)

sl - Select Mailbox (IMAP only). Attributes include:

- Common: ma, si, tr, us

Chapter 31 Message expiration

Message expiration automatically removes messages from the Message Store based on configured criteria. For example, automatic expiration can selectively remove old messages, overly large messages, seen or deleted messages, messages with specific Subject: lines, messages of a certain type, *etc.*.

Note: Oracle Communications Messaging Server removes messages without giving a warning, so it is important to **inform users about message expiration policies**! Unexpected message removal can be a source of consternation for users and administrators.

Message expiration is performed by the `imexpire` utility. This utility may be run manually, and/or be configured to run automatically on a schedule as configured for the Scheduler's expire task.

The rules for what the `imexpire` utility (the expire task) does -- which messages it checks, for which expiration criteria -- may be controlled in two distinct ways. An older, and nowadays less preferred, method is to define global rules (applying to all messages and users) via Message Store expirerule options (corresponding to legacy configuration configutil parameters). The preferred method is to define the expiration rule via `store.expirerule` files.

For use of `imexpire` post-message-delivery to scan messages for spam or viruses, see `imexpire` invoking spamfilter pacakges.

31.1 `store.expirerule` files

The use of `store.expirerule` files (rather than Message Store `expirerule` options) is the preferred way to configure message expiration rules for the Message Store.

Multiple `store.expirerule` files may be configured, each located in the directory that pertains to the scope of the expiration rules. That is, rules that apply globally to the entire Message Store are located in one directory, rules that apply to a particular partition are located in a directory for that partition, while rules that apply to specific users are located in user-specific directories, which may be further scoped as applying only to a particular folder of that user. (This feature of scoped expiration rules is one of the benefits of using `store.expirerule` files, rather than Message Store `expirerule` options, as such options are inherently global and apply to the whole Message Store.)

1. The global message expiration rules are stored in the *`CONFIGROOT`*`/store.expirerule` file. (Note that each global rule will be checked against *every* user's *every* folder, potentially resulting in noticeable processing overhead depending upon the number of global rules and the number of users and folders. Therefore, in general, avoid putting any more expiration rules than necessary in this file; in particular, do not put partition, user, or folder rules in this global rules file.)

2. Partition message expiration rules are stored in the directory location specified by the corresponding partition's `partition:`*`partition-name`*`.path` value, so typically

DATAROOT/store/partition/*partition-name*/store.expirerule

3. User message expiration rules are stored as a:

   DATAROOT/store/partition/*partition-name/userid*/store.expirerule file.

   Alternatively, user message expiration rules may be stored in a partition store.expirerule file, but defined to apply to a user via the file's setting of a ruleset folderpattern attribute to a value user/*userid*.*

4. Folder-specific message expiration rules are stored as a:

   DATAROOT/store/partition/*partition-name/userid/folder*/store.expirerule

   file. Alternatively, user message expiration rules may be stored in a partition store.expirerule file, but defined to apply only to a folder via the file's setting of a ruleset folderpattern attribute to a value user/*userid/folder-name*

It is also possible to exclude specified users from the expiration rules via use of a expire_exclude_list file, located in the *CONFIGROOT* directory. (For instance, an administrator, postmaster, or archival address might be exempted from message expiration via this mechanism.) The format of the file is line oriented; the user ID of each user to be exempted from message expiration should be listed, one user ID per line of the file.

When imexpire runs, it will create one thread per partition; each partition thread will go through the list of user folders under its assigned partition, loading the relevantly scoped store.expirerule files as it goes. It checks each folder against the expiration rules applicable to that folder, and expunge messages as needed.

31.1.1 store.expirerule file rulesets

As of MS 6.2p4, the rules in a store.expirerule file can be grouped into *rule sets* by prefixing the attribute names with a textual *rule name*. For instance:

```
Rule1.regexp: 1
Rule1.folderpattern: user/.*
Rule1.messagedays: 30
Rule2.folderpattern: user/.*@example.org/.*
Rule2.messagedays: 15
```

The attributes which may be specified in a rule set are listed in imexpire Ruleset Attributes.

Table 31.1 imexpire Ruleset Attributes

Attribute	expirerule Option	Description (Attribute Value)
action	*None; expirerule options always operate as discard rule*	Specifies an action to perform on the messages matching the expiration rules. The possible values are: • discard -- discards the message. This is the default. • report -- prints the mailbox name, uid-validity, and uid to stdout. The message is not changed by this action. • archive -- archives the message with the Compliance and Content Management System. The message is not changed by this action. This is a form of operational archiving, and is no longer supported as of MS 8.0.2. • fileinto: *folder-name* -- files the message into the specified folder. The shared folder prefix can be specified to file messages to folders owned by another user. The original message is discarded if the fileinto operation is successful.

exclusive	exclusive	Specifies whether or not this is an exclusive rule. The default is 0 (this rule is not exclusive) If specified as exclusive, then only this rule applies to the specified mailbox(es) and all other rules are ignored. If more than one exclusive rule exists, the last exclusive rule loaded will be used. For example, if a global and a local exclusive rule are specified, the local rule will be used. If there is more than one global exclusive rule, the last global rule listed by configutil is used. (1 or 0)
expires		imexpire will select the message if the date value specified by an Expires: header field value is older than the expiration date based on the messagedays attribute value. If multiple expiration header fields are specified, the earliest expiration date will be used. (string)
expiry-date		imexpire will select the message if the date value specified by an Expiry-date: header field value is older than the expiration date based on the messagedays attribute value. If multiple expiration header fields are specified, the earliest expiration date will be used. (string)
folderpattern	folderpattern	Specifies the folders affected by this rule. The format must start with user/, which represents the directory *DATAROOT*/store/partition/*/. See imexpire folder patterns. (POXIX regular expression)
messagecount	messagecount	Maximum number of messages in a folder. Oldest messages are expunged as additional messages are delivered. (integer)
foldersizebytes	foldersizebytes	Maximum size of folder, in bytes, before the oldest messages are expunged when additional messages are delivered. (integer in bytes)
messagedays	messagedays	Number of days in the Message Store before message is expunged. (integer)
messagesize	messagesize	Maximum size of message, in bytes, before it is marked to be expunged. (integer in bytes)
messagesizedays	messagesizedays	Grace period: days an over-sized message should remain in a folder. (integer)
messageheader.*field-name*		Specifies a header field name and header field value string. The header field name and header field values are not case-sensitive, and regular expressions are not recognized. Example: Rule1.messageheader.Subject: Get Rich Now! Headers other than Subject: can be used. (string)
regexp		Enable UNIX regular expression in rules creation. If not set, IMAP expressions will be used. See imexpire folder patterns for further details. (0 or 1)
savedays		Number of days a message is saved in a folder until being expunged. (integer)
seen	seen	\Seen is an IMAP system flag set by the Message Store when the user opens a message. If the store.expirerule file rule set attribute seen is set to and, then the message must be seen *and* another rule criterion must be met before the rule is fulfilled. If the attribute seen is set to or, then a message only needs to be seen *or* another rule criterion be met before the rule is fulfilled. The default is and. (and or or)
sieve		A Sieve test specifying message selection criteria. Example: Rule17.sieve: header :contains "Subject" "Make money fast" In order for sieve attribute values to take effect, the expiresieve Message Store option must be enabled; they will be ignored otherwise. Note that by also using a rule set action of fileinto: *folder-name* in a rule set with a sieve test, an effect of performing a Sieve "fileinto" based on a Sieve test can be obtained; that is, the rule set's sieve value supplies the Sieve test and the rule set's action: fileinto: supplies the fileinto folder name. (string)
deleted	deleted	\Deleted is an IMAP system flag set by the Message Store when the user deletes a message. If the store.expirerule file rule set attribute deleted is set to and, then the message must be deleted *and* another rule criterion must be met before the rule is fulfilled. If the attribute deleted is set to or, then a message only needs to be deleted *or* another rule criterion be met before the rule is fulfilled. The default is and. (and or or)
join		(New in MS 7.0.4) join may only be specified(set 1) when exclusive rule is set . The default is 0. Specifying join: 1 means all rule criterias are combined as single rule. (0 or 1)
userflag.*flag-name*		(New in MS 7.0.5, as well as MS 7.0.4.24) Valid values are 'and' and 'or'; the default is 'and'. Specify user IMAP flags in expiration rules. For instance, the following rule expires those messages with the 'junk' flag set, and which are older than 30 days: messagedays: 30 userflag.junk: and
channel		(New in MS 7.0.5) Specify the name of an MTA channel as which to "run" for purposes of spam/virus filtering. In order for a channel attribute value to take effect, the expiresieve Message Store option must be enabled. (string)
rescanhours		(New in MS 7.0.5) When using imexpire to perform post-delivery spam/virus filtering, the rescanhours tells imexpire to rescan those message that have not been scanned for the specified number of hours. The default is 10. In order for any rescanhours attribute value to take effect, the expiresieve Message Store option must be enabled. (integer)

Note that discarded messages are only expunged at the end of the imexpire run, that is, only one expunge operation is performed on each folder.

31.2 imexpire folder patterns

Folder patterns can be specified using POSIX regular expressions, if the imexpire attribute regexp is set to 1. If the regexp attribute is not set (the default), IMAP expressions will

be used. The format is that the folder pattern must start with `user/`, followed by a pattern. imexpire folder pattern examples shows the both formats of folder pattern for various folders.

Table 31.2 imexpire folder pattern examples

Scope	Folder patterns (regexp: 0)	Folder patterns (regexp: 1)
Apply rule to all message in all folders of *userid*.	user/*userid*/*	user/*userid*.*
Apply to messages of *userid* in folder Sent.	user/*userid*/Sent	user/*userid*/Sent
Apply rule to entire Message Store.	user/*	user/.*
Apply rule to any folder named Trash, anywhere in any user's hierarchy.	user/*/Trash	user/.*/Trash
Apply rule to folders of users in hosted domain example.org.	user/*@example.org/*	user/.*@example.org/.*
Apply rule to folders of users in the default domain.	*Not applicable*	user/[^@]*/.*

31.3 imexpire and localized mailbox names

The IMAP protocol specifies that mailbox names use modified UTF-7 encoding. Messaging Server supports localized character sets on external interfaces so that mailbox names can be localized. Internally, however, the system converts the localized name to MUTF-7. Thus, a folder that has a localized mailbox name on a client will have a corresponding mailbox file name in MUTF-7. (Note that IMAP error messages will output mailbox names in MUTF-7 and not the localized character set.)

In general, most Message Store utilities that require mailbox names expect the names in the localized character set, although they may have an option flag that allows a different character set to be used. These utilities include `reconstruct`, `mboxutil`, `imsbackup`, `imsrestore`, and `hashdir`. However, `imexpire` requires that the mailbox name, specified as the attribute `folderpattern`, be in MUTF-7. Using a localized name will not work.

To obtain the appropriate `folderpattern` value for use with `imexpire`, it may be necessary to convert a localized mailbox name to the modified UTF-7 equivalent. This can be done using the `mboxutil -E` command as follows:

```
$ mboxutil -l -p user/han/*

  msgs  Kbytes last msg         partition   quotaroot mailbox

    57     100 2010/04/29 11:18 primary       5242880 user/han/INBOX
     1       1 2010/04/30 12:56 primary             -
user/han/multibyte-mailbox-name

$ mboxutil -l -E MUTF-7 -p user/han/*

    57     100 2010/04/29 11:18 primary       5242880 user/han/INBOX
     1       1 2010/04/30 12:56 primary             -
user/han/&kAFP4W4IMH8wojCkMMYw4A-
```

The output of the first `mboxutil` command shows the localized mailbox name. The output of the second `mboxutil` command shows the mailbox name in MUTF-7. The MUTF-7 mailbox name is identical to that shown in response to an IMAP LIST command:

```
x list "" *
```

```
* LIST (\NoInferiors) "/" INBOX
* LIST (\HasNoChildren) "/" &kAFP4W4IMH8wojCkMMYw4A-
```

To convert the local charset to modified UTF-7 encoding, use the mboxutil command's -E option:

```
$ mboxutil -l -E MUTF-7 -P user/han/multibyte-mailbox-name

  msgs  Kbytes last msg         partition   quotaroot mailbox

     1       1 2010/04/30 12:56 primary             -
user/han/&kAFP4W4IMH8wojCkMMYw4A-
```

Such a mboxutil -E command can be used in preparation for use of any utility that requires the use of an MUTF-7 mailbox name, including as discussed here in preparation to use imexpire.

Chapter 32 Store Index and search

The message store supports three index and search services: ISS, DSE Solr, and Elasticsearch. Note that DSE Solr is no longer available starting with the Messaging Server 8.1 release. Classic store supports ISS and Elasticsearch. DSE Cassandra store supports Solr and Elasticsearch. Open source Cassandra store supports Elasticsearch. The classic message store does not enable any index and search services by default. When index and search service is not enabled, IMAP uses brute force for body search. Brute force message body search is very I/O intensive. Consider using an index and search service when search performance is crucial.

When either Elasticsearch or DSE Solr is enabled, the message store uses ISC to convert binary message content to text before indexing. Therefore, ISC must be configured on when either Elasticsearch or DSE Solr is enabled. ISS has built-in content conversion, so ISC is not required.

ISC

Indexed Search Converter (ISC) is a Java and http-based server that converts message content to text for index and search purposes. ISC is included in the Messaging Server package and uses Apache Tika to detect and extract text content from email messages. The converted texts are cached. On indexing, the message store looks up the text content in the cache. If the text is found, the message store reads it from the cache. Otherwise, the message store sends the meta data to ISC. ISC reads the message content, converts it to text, and writes the text to the cache.

In classic store, ISC uses the file system to cache the text content. In Cassandra store, ISC uses Cassandra to store the cache content.

ISC requires read access to the message store in order to convert the content of a message. When used with classic store, ISC can be co-located with other message store processes or can run on a separate server with shared filesystem access to message store data.

Review the ISC Options for more information about configuring ISC.

Elasticsearch

Elasticsearch is a distributed search engine based on Lucene. It provides full text search services with built-in high availability, replication, horizontal scaling, and automatic load balancing. When Elasticsearch is enabled, the message store sends the message content to Elasticsearch for indexing when messages are appended to the mailboxes; the IMAP server sends search queries to Elasticsearch to perform textual searches.

To enable elasticsearch:

- set `store.searchengine` to `elastic`
- set `elasticsearch.hostlist` to a list of Elasticsearch hosts
- set `elasticsearch.numshards` to the number of shareds
- set `elasticsearch.numreplicas` to the number of replicas

- set `elasticsearch.storesource` to false if storage space is limited

Review the Elasticsearch options for more information about configuring use of Elasticsearch.

DSE Solr

Datastax Enterprise (DSE) Solr is an index and search component provided by Datastax. DSE Solr index and search is enabled by default when Cassandra message store is enabled prior to Messaging Server 8.1. Messaging Server provides the FIT plugin to prepare the message content for indexing. FIT must be installed on all the Solr nodes.

ISS

Oracle Communications Indexing and Search Service (ISS) is a separate index and search service developed by Oracle. The message store uses Glassfish MQ event notification services to send data for indexing to ISS and ISS re-synchronizes with the Message Store over IMAP. Starting with Messaging Server 8.0.2.2, ISS is deprecated and support for ISS may be removed in a future release of Messaging Server. Starting with Messaging Server 8.1, ISS is no longer included with Messaging Server and use of a previous version of ISS is only supported for the purpose of migrating from ISS to Elasticsearch.

Review the Indexer Options for more information about configuring use of ISS.

32.1 Migrating from ISS to Elasticsearch

The message store supports the following migration strategies:

Side by side migration

- Install a new message store.
- Enable Elasticsearch and ISC on the new message store.
- Migrate users from the old store to the new store with rehostuser.
- Decommission the old store when all the users are migrated to the new store.

In-place migration

- Upgrade the messaging server to 8.0.2.2.
- Enable Elasticsearch and ISC.
- Enable impurge if it is not enabled.
- Restart the server.
- Folders are scheduled for migration automatically when they are accessed.
- Run 'imcheck -H' to migrate all the folders.
- Remove ISS event notification target and ISS client configuration on the message store.

32.2 Migrating from DSE SOLR to Elasticsearch

The message store supports the following migration strategies:

Side by side migration

- Install an Elasticsearch cluster.
- Install a Cassandra cluster.
- Install a new message store.
- Enable Elasticsearch, Cassandra and ISC on the new message store.
- Migrate users from the old store to the new store with rehostuser.
- Decommission the old store when all the users are migrated to the new store.

In-place migration

- Install an Elasticsearch cluster.
- Upgrade the messaging server to the latest version.
- Enable Elasticsearch.
- Enable ISC and impurge if it is not enabled.
- Configure `store.solrconnectpoints` if it is not configured
- Restart the server.
- Folders are scheduled for migration automatically when they are accessed.
- Run 'imcheck -H' to migrate all the folders.
- Remove Solr configuration on the message store.
- Disable Solr in DSE cluster, or remove the Solr DC and update the mbox keyspace accordingly

32.3 Search Technology Comparison

Comparisons can be made based on various different criteria. The following sections cover some of the possibilities.

32.3.1 Substring vs. Indexed Search

The classic search technology built-in to imapd performs a substring search (a search term matches any word containing that substring). The indexed search technologies (Elasticsearch, DSE Solr, and ISS) use a word-based search where a search term must match an entire word (or root word). Wildcard options for indexed search are available, but add significant overhead so searches take longer and require more resources. It may be desirable to configure wildcard use for address headers. In order to strictly comply with the IMAP standard for searching, all textual searches must be configured as substring searches (using the `imap.indexer.substring_search` option). However, modern web searches tend to be

word-based so our product defaults to word-based searches for message text as we believe that behavior will be faster and familiar to end-users.

32.3.2 Search Complexity

Elasticsearch supports all searches that IMAP supports (flag and annotation terms in an IMAP search command are converted to UID lists by imapd). DSE Solr supports all searches; however the ANNOTATE extension (RFC 5257) is not implemented on Cassandra Store. ISS only supports a subset of search operations; see the `imap.indexer.enable` option for a description of ISS limitations.

32.3.3 Index Storage

Elasticsearch can store indexes on Elasticsearch local disk without the need for a storage array. Use of local SSD will reduce the number of nodes required in the Elasticsearch cluster for a given workload. Use of local SSD is recommended for DSE Solr search. Use of a storage array with fast iSCSI support is recommended for ISS.

32.3.4 Search Host Failures

The imapd Elasticsearch client will round-robin between hosts listed in the `elasticsearch.hostlist` option; it will periodically retry Elasticsearch hosts that were previously down. The message store ISC client will failover to a backup host if more than one host is listed in the `isc_client.ischosts` option; otherwise conversion will be deferred and performed by `impurge` later. The DSE Solr client will use the `store.solrconnectpoints` option to bootstrap a list of available DSE Solr servers; and DSE Solr uses the Cassandra Gossip protocol to stay current on the list of available Cassandra/Solr nodes as long as at least one of the nodes listed in `solrconnectpoints` is online when the server process starts. When the ISS host is unavailable, search will be performed by classic search. With ISS, it is important to keep your JMQ broker running reliably.

32.3.5 High Availability

Elasticsearch has built-in high availability; a minimum cluster size of three nodes is recommended. DSE Solr search uses Cassandra's high availability mechanism. A minimum cluster of three Cassandra-only and two Cassandra/Solr nodes is recommend for the msgindex table. ISS has no HA mechanism; a reliable filesystem is recommended and fallback to classic search occurs when ISS fails.

32.3.6 Data Growth

Elasticsearch and DSE Solr scale horizontally; as your data size grows you will need to add servers to your Elasticsearch or Cassandra clusters. For ISS, once the capacity of a message store or ISS server is exceeded; you must use the `rehostuser` tool to move users to a new message store with a new ISS server.

32.3.7 Reindexing

Elasticsearch will require a reindex when a classic store userid is renamed. However, rehostuser and renaming mailboxes will not require a reindex with Elasticsearch as long as source and target hosts share the same Elasticsearch cluster. Cassandra store supports separate

external and persistent userids so DSE Solr will not require a reindex when the external userid is changed. ISS requires a reindex of impacted content when a userid is renamed, rehostuser is used, or a mailbox is renamed.

32.3.8 Stop Words

In order to reduce the size of the index, English stop words are not included in the index by default for Elasticsearch and DSE Solr search. Stop words in a search query will match no messages if they are part of a single search key or are in a boolean OR clause. For Elasticsearch, stop words in an AND clause will match all messages. Stop words used in a wildcard search will only match text containing a word that contains the stop word as a substring.

32.3.9 Whitespace

The classic search ignores whitespace, so search terms will match across whitespace breaks. Whitespace is significant to indexed search technologies.

32.3.10 Punctuation

The classic search is punctuation sensitive. The indexed search technologies treat most punctuation as whitespace equivalent.

32.3.11 Diacritical Sensitivity

The classic search is case and diacritical insensitive by default. The `diacritical_sensitive_languages` option can be used to make classic search diacritical sensitive for certain languages. The indexed search technologies are diacritical sensitive by default.

32.3.12 Convergence/ISS Attachment Search

The attachment search feature is only available with ISS which is deprecated. If your deployment depends on this feature please contact Oracle support and explain how you use it to assist Oracle's determination on whether to implement an equivalent feature with Elasticsearch.

32.3.13 Non-IMAP Search API

ISS supports a non-IMAP search API via an HTTP-based protocol. At this time, Oracle reserves the right to make incompatible changes to the elasticsearch and DSE/Solr indexing models if that is necessary to improve the product (the odds of such changes will decrease over time). If you have a business use case that requires an HTTP-based search API please contact Oracle support for consideration of such a request.

32.4 Elasticsearch options

Elasticsearch options control IMAP search operations when the `searchengine` is set to `elastic`. See also the ISC Options and Indexer Options.

32.4.1 `hostlist` Option Under elasticsearch

The hostlist elasticsearch option specifies a space separated list of elasticsearch hosts. Host format is *host*[:*port*]. If *port* is not specified, the port number is determined based on the setting of the elasticsearch.port option.

32.4.2 port Option Under elasticsearch

The port elasticsearch option specifies the default elasticsearch server TCP port number. This is used if no port is specified in the elasticsearch.hostlist option. The default is 9200.

32.4.3 authusername Option Under elasticsearch

The authusername Elasticsearch option sets the username that will be used for HTTP basic authentication when communicating with Elasticsearch. The elasticsearch.authpassword option must also be set. This option is only used when the store.searchengine option is set to elastic.

32.4.4 authpassword Option Under elasticsearch

The authpassword Elasticsearch option sets the password that will be used for HTTP basic authentication when communicating with Elasticsearch. The elasticsearch.authusername option must also be set. This option is only used when the store.searchengine option is set to elastic.

32.4.5 sslusessl Option Under elasticsearch

The sslusessl Elasticsearch option will require use of SSL when communicating with Elasticsearch. This option is only used when the store.searchengine option is set to elastic.

32.4.6 storesource Option

The _source field in Elasticsearch contains the document body that was passed at index time. If elasticsearch.storesource is enabled, the message store will create the elasticsearch store/msg index mapping with the _source field enabled; IMAP copy will use the _source field from Elasticsearch to index the message on the destination folder. The _source field data consumes a lot of disk space. You might want to disable the _source field if storage space is limited. Disabling the _source field will disable the ability to reindex, upgrade or repair index from Elasticsearch. See https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-source-field.html before disabling this option.

32.4.7 numshards Option

The numshards field in Elasticsearch specifies the number of shards in the Elasticsearch message store index. The elasticsearch.numshards is used by stored to create the message store index in Elasticsearch. The number of shards cannot be changed after the index is created.

32.4.8 numreplicas Option

The numreplicas field in Elasticsearch specifies the number of replicas in the Elasticsearch message store index. The elasticsearch.numreplicas is used by stored to create the

elasticsearch message store index. The message store does not update the number of replicas after the index is created. The number can be updated in Elasticsearch manually.

32.5 Indexer options

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`.

32.5.1 `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.

32.5.2 `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. The default is `8070`.

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.

32.5.3 `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.

32.5.4 timeout Option Under indexer

The timeout Indexer option specifies the timeout in seconds for read and write operations between the IMAP server and ISS.

32.5.5 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.

32.5.6 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.

32.5.7 substring_search Option

The substring_search Indexer option specifies which, if any, search parameters sent to an external searchengine, 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 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 the search engine.

When the internal search engine is used, a substring match is always performed and this option is ignored.

32.5.8 suffix_search Option

The suffix_search Indexer option specifies which, if any, search parameters sent to an external searchengine should be followed by a "*", so that a search for subject book would match "bookmark" as well as "book". The search parameter will not be followed 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 header separated by one or more spaces. The default is the empty string.

(Note that the meanings of suffix_search and prefix_search could be considered reversed; read carefully before setting.)

When the internal search engine is used, a substring match is always performed and this option is ignored.

32.5.9 prefix_search Option

The prefix_search Indexer option specifies which, if any, search parameters sent to an external search engine 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 header separated by one or more spaces. Use with caution as this imposes a significant load on the search engine. The default is the empty string.

(Note that the meanings of suffix_search and prefix_search could be considered reversed; read carefully before setting.)

When the internal search engine is used, a substring match is always performed and this option is ignored.

32.5.10 sslusessl Option Under indexer

If the sslusessl Indexer option is set to 1, then the IMAP server uses SSL to authenticate to the ISS, connecting to the port specified by the port Indexer option (service.imap.indexer.port option in legacy configuration).

32.6 ISC options

ISC Options control the Indexed Search Converter process. The ISC process is used to extract text from messages for use by an indexed search engine. It is presently required when your searchengine is set to dse or elastic.

32.6.1 enable Option Under isc

The enable ISC option, (isc.enable in Unified Configuration, not available in legacy configuration), enables the ISC service on start-msg startup.

This option defaults to 0 if not set, but initial configuration may enable the option as appropriate.

32.6.2 authpassword Option Under isc

The authpassword ISC option sets the password that will be used by the Message Store to communicate with the ISC server and will be used by the ISC server for server-to-server authentication. This option is only applicable when use of the Indexed Search Converter is enabled.

32.6.3 basicjavaswitches Option

The basicjavaswitches ISC option specifies a space-separated list of JVM options (heap settings, GC options, etc) to use when starting the java isc process. The default settings for this option are the recommended JVM switches for running ISC, described below. The recommended JVM switches may change between releases as Oracle gains tuning experience with ISC. Unless the customer determines one of the recommended JVM switches is inappropriate for their environment changes to this option are discouraged as an explicit setting means future changes to the recommended defaults will not be applied.

The extrajavaswitches option can be used to append customer preferred JVM switches without replacing the recommended default settings.

```
## Heap options
# Sets the initial size (in bytes) of the heap.
-Xms8g

# Specifies the maximum size (in bytes) of the heap.
-Xmx32g

# Specifies the recommended garbage collector for ISC
-XX:+UseG1GC

# Specifies the maximum pause for garbage collection
-XX:MaxGCPauseMillis=200
```

Changes to this option do not take effect until ISC is restarted.

32.6.4 cachettl Option Under isc

The cachettl option specifies the time-to-live, measured in seconds, for the converted text in the conversion cache. The converted contents expire after this time, and ISC will have to re-convert any new document with the same content. The default is 30 days. The minimum is one hour.

32.6.5 extrajavaswitches Option

The extrajavaswitches ISC option specifies a space-separated list of JVM options that are used in addition to the recommended Java switches present in the basicjavaswitches ISC option. This option is for customer-provided JVM modifications while basicjavaswitches should only be used to override the recommended JVM switches. For a full list of available JVM options, please refer to the Java official documentation at http://docs.oracle.com.

Changes to this option do not take effect until ISC is restarted.

32.6.6 sslusessl Option Under isc

Flag to enable SSL for ISC server connections. SSL is disabled by default. If this option is set to 1, then the corresponding isc_client.sslusessl option must also be set to 1 on the LMTP host(s), and the fit.sslusessl option must be set to 1 on the Cassandra host(s) that connect to this ISC server.

If this option is set to 1, then the base.ssljkspath and base.ssljkspassword options must be set to the Java SSL keystore path and the Java keystore password respectively.

32.6.7 authusername Option Under isc

The authusername ISC option sets the username that will be used by the Message Store to communicate with the ISC server and will be used by the ISC server for server-to-server authentication. This option is only applicable when use of the Indexed Search Converter is enabled.

32.6.8 server_port Option Under isc

The server_port isc option specifies the Http port for the ISC process. If empty or not set, the default is 8070.

If isc.sslusessl is set to 1, this will be setup as an https port.

32.6.9 maxthreads Option Under isc

The maxthreads isc option specifies the maximum number of parallel requests that the isc http server can process. Default is 10000. Cannot exceed 60000.

Starting with 8.0.2.2, this has no effect and may be removed in a future release.

32.6.10 logdir Option Under isc

The logdir isc option specifies the log files location for ISC.

The user that owns the ISC process must have write permissions to this location.

32.6.11 authusername Option Under isc_client

The authusername isc_client option sets the username that will be used by the LMTP server to communicate with the ISC server and will be used by the ISC server for server-to-server authentication. This option is only applicable when use of the Indexed Search Converter is enabled.

32.6.12 authpassword Option Under isc_client

The authpassword ISC_CLIENT option sets the password that will be used by the LMTP server to communicate with the ISC server for authentication. This option is only applicable when use of the Indexed Search Converter is enabled.

32.6.13 ischosts Option Under isc_client

The ischosts isc_client option specifies a space-separated list of hosts (with optional :port) for the isc server. List multiple hosts to avoid a single point of failure.

32.6.14 max_conns Option Under isc_client

The max_conns isc_client option specifies the maximum number of connections that are permitted from a single LMTP process to the ISC server. Starting with 8.0.2.2, this option is preferred over the isc_client.connlimits option.

32.6.15 sslusessl Option Under isc_client

Flag to enable SSL for ISC server connections. SSL is disabled by default. If this option is set to 1, then the corresponding isc.sslusessl option must also be set to 1 on the host(s) specified by isc_client.ischosts.

32.6.16 server_port Option Under isc_client

The `server_port isc_client` option specifies the ISC server port number. If empty or not set, the default is `8070`.

The value for this option must match the isc.server_port on the host(s) specified by isc_client.ischosts.

32.7 FIT options

32.7.1 `logdir` Option Under fit

The `logdir` fit option specifies the log files location for FIT.

The user that owns the DSE (Cassandra) process must have write permissions to this location.

32.7.2 `jloglevel` Option Under fit

The `jloglevel` stats option specifies the FIT log level. Default is INFO. Available options (in increasing order of importance) are FINEST, FINER, FINE, INFO, WARNING and SEVERE.

Chapter 33 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.

See also the `connlimits` option, which may be used to limit the number of connections, (rather than outright blocking). And see the `bg*` options, such as `bgpenalty`, which may be configured to penalize failed authentication attempts.

For allowing inspection of (rather than controlling/limiting) when a user last accessed the Message Store, see the `enablelastaccess` base option. For allowing inspection of what users are currently connected via the IMAP server or via the MSHTTP server, see the `imap.enableuserlist` and `http.enableuserlist` options. And see the `imsconnutil` utility for displaying such information.

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 34 IMAP options

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 logfile options, 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.

34.1 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.

34.2 actions Option

The actions option (available for imap, pop, and messagetrace) specifies the actions enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(a1 a2 a3)" to enable only attributes listed, or "-(a1 a2 a3)" to enable all attributes except those listed. See the Store Transaction Log Format section for permitted attribute codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(fe)" (log everything except the fetch action) by default.

34.3 actionattributes Option

The actionattributes option (available for imap, pop, and messagetrace) specifies the action attributes enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(e1 e2 e3)" to enable only event codes listed, or "-(e1 e2 e3)" to enable all events except those listed. See the Store Transaction Log Format section for permitted event codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(mi)" (log all attributes except the message id attribute) by default. Note that the message id is expensive to extract from message headers and requires per-message logging of expunge events.

34.4 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.

34.5 allowanonymouslogin Option Under imap

The allowanonymouslogin IMAP option enables the SASL ANONYMOUS mechanism for use by IMAP.

34.6 authfaildelay Option

The authfaildelay option (available under IMAP and POP) determines how long the server delays before reporting an authentication failure. This option is present only for the back-end POP3 and IMAP servers. The MMP uses a more sophisticated badguyslist facility (see bgpenalty). Decreasing the authfaildelay option below the default value (3) is not recommended.

34.7 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).

34.8 bgmax Option

The bgmax option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See bgpenalty for more information.

34.9 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.

The bgpenalty option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the length of time in seconds added to the authentication delay after each failed authentication.

34.10 bgmaxbadness Option

The bgmaxbadness option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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.

34.11 bgdecay Option

The bgdecay option (available under base, imap, pop, mmp, imapproxy, and popproxy) represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See bgpenalty for more information.

34.12 bglinear Option

The bglinear option (available under base, imap, pop, mmp, imapproxy, and popproxy) defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See bgpenalty for more information.

34.13 bgexcluded Option

The bgexcluded option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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).

34.14 broken_client_defer_exists Option

Starting in Messaging Server 8.0, the IMAP server notifies the client as soon as new mail arrives regardless of whether the IDLE command is in use. The IMAP specification requires clients to support this behavior (RFC 3501 section 2.2.2). Some clients fail to comply with the standard and lose state information about new messages. This option can be set to workaround clients that are broken in this way. It causes IMAP EXISTS and RECENT responses to be deferred unless a command is in progress.

34.15 capability_acl Option

The capability_acl IMAP option, when set to 1 (the default), causes the IMAP server to enable the ACL IMAP extension. (See RFC 4314 (IMAP4 Access Control List (ACL) Extension).)

34.16 capability_annotate Option

The capability_annotate IMAP option, when set to 1 (the default), causes the IMAP server to enable the ANNOTATE-EXPERIMENT-1 IMAP extension. (See RFC 5257 (IMAP ANNOTATE Extension).)

34.17 capability_binary Option

The capability_binary IMAP option, when set to 1 (the default), causes the IMAP server to enable the BINARY IMAP extension. (See RFC 3516 (IMAP4 Binary Content Extension).)

34.18 capability_catenate Option

The capability_catenate IMAP option, when set to 1 (the default), causes the IMAP server to enable the CATENATE IMAP extension. (See RFC 4469 (IMAP CATENATE Extension).)

34.19 capability_children Option

The capability_children IMAP option, when set to 1 (the default), causes the IMAP server to enable the CHILDREN IMAP extension (See RFC 3348 (IMAP4 Child Mailbox Extension).)

34.20 capability_condstore Option

The capability_condstore IMAP option, when set to 1 (the default), causes the IMAP server to enable the CONDSTORE IMAP extension. (See RFC 4551 (IMAP Extension for Conditional STORE Operation or Quick Flag Changes Resynchronization).)

Note that the QRESYNC extension subsumes all of CONDSTORE. So if capability_qresync is enabled, CONDSTORE is effectively enabled regardless of whether or not it is enabled due to capability_condstore.

34.21 capability_context_search Option

The capability_context_search IMAP option, when set to 1 (the default), causes the IMAP server to enable the CONTEXT=SEARCH IMAP extension. (See RFC 5267 (Contexts for IMAP4).)

34.22 capability_context_sort Option

The capability_context_sort IMAP option, when set to 1 (the default), causes the IMAP server to enable the CONTEXT=SORT IMAP extension. (See RFC 5267 (Contexts for IMAP4).)

34.23 capability_create_special_use Option

The capability_create_special_use IMAP option, when set to 1 (the default), causes the IMAP server to enable 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).

34.24 capability_enable Option

The capability_enable IMAP option, when set to 1 (the default), causes the IMAP server to enable the ENABLE IMAP extension. (See RFC 5161 (The IMAP ENABLE Extension).)

34.25 capability_esearch Option

The capability_esearch IMAP option, when set to 1 (the default), causes the IMAP server to enable the ESEARCH IMAP extension. (See RFC 4731 (IMAP4 Extension to SEARCH Command for Controlling What Kind of Information Is Returned).)

See also the capability_multisearch and maxsearchmailboxes IMAP options.

34.26 capability_esort Option

The capability_esort IMAP option, when set to 1 (the default), causes the IMAP server to enable the ESORT IMAP extension. (See RFC 5267 (Contexts for IMAP4).)

34.27 capability_id Option

The capability_id IMAP option, when set to 1 (the default), causes the IMAP server to enable the IMAP ID extension. (See RFC 2971 (IMAP4 ID extension).)

34.28 capability_idle Option

Setting the capability_idle IMAP option to 1 (the default) causes the IMAP server to enable the IDLE IMAP extension, if ENS is also enabled. (See RFC 2177 (IMAP4 IDLE command).)

34.29 capability_imap4 Option

Enable the IMAP4 capability. The default is normally 0 (false), but the default is 1 (true) if obsoleteimap (in legacy configuration, local.obsoleteimap) is set. Compare with capability_imap4rev1, which enables the modern IMAP protocol. (For the definition of IMAP4, see RFC 1730 (INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4). The more modern version of IMAP is IMAPv4rev1, initially defined in RFC 2060 (INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1), since updated by RFC 3501 (INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1).)

34.30 capability_imap4rev1 Option

The capability_imap4rev1 IMAP option, when set to 1 (the default), causes the IMAP server to enable the IMAP4rev1 capability. (See RFC 3501 (INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1).)

34.31 capability_language Option

The capability_language IMAP option, when set to 1 (the default), causes the IMAP server to enable the LANGUAGE IMAP extension. (See RFC 5255 (Internet Message Access Protocol Internationalization).)

34.32 capability_list_status Option

The capability_list_status IMAP option, when set to 1 (the default), causes the IMAP server to enable 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).

34.33 capability_literal Option

The capability_literal IMAP option, when set to 1 (the default), causes the IMAP server to enable the LITERAL+ IMAP extension. (See RFC 2088 (IMAP4 non-synchronizing literals).)

34.34 capability_login_referrals Option

The capability_login_referrals IMAP option, when set to 1 (the default), causes the back-end IMAP server to enable 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. (See RFC 2221 (IMAP4 Login Referrals).)

34.35 capability_metadata Option

The capability_metadata IMAP option, when set to 1 (the default), causes the IMAP server to enable the METADATA IMAP extension. (See RFC 5464 (The IMAP METADATA Extension).)

34.36 capability_multisearch Option

For 7.0.5.31.0 and later, the capability_multisearch IMAP option is disabled by default and if enabled will enable the experimental XMSEARCH IMAP extension that is a subset of the functionality described in RFC 6237 (IMAP4 Multimailbox 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 7377 (IMAP4 Multimailbox SEARCH Extension).

See also the capability_esearch and maxsearchmailboxes IMAP options.

34.37 capability_namespace Option

The capability_namespace IMAP option, when set to 1 (the default), causes the IMAP server to enable the NAMESPACE IMAP extension. (See RFC 2342 (IMAP4 Namespace).)

34.38 capability_notify Option

The capability_notify IMAP option, when set to 1, causes the IMAP server to enable the NOTIFY IMAP extension. The default is 0. (See RFC 5465 (The IMAP NOTIFY Extension).)

34.39 capability_qresync Option

The capability_qresync IMAP option, when set to 1 (the default), causes the IMAP server to enable the QRESYNC IMAP extension. (See RFC 7162 (IMAP Extensions: CONDSTORE and QRESYNC).)

Note that the QRESYNC extension subsumes all of the CONDSTORE extension. So if capability_qresync is enabled, CONDSTORE is effectively enabled regardless of whether or not it is enabled due to capability_condstore.

34.40 capability_quota Option

The capability_quota IMAP option, when set to 1 (the default), causes the IMAP server to enable the QUOTA IMAP extension. (See RFC 2087 (IMAP4 QUOTA extension).)

34.41 capability_savedate Option

The capability_savedate IMAP option, when set to 1 (the default), causes the IMAP server to enable the SAVEDATE IMAP extension. (See RFC 8514.)

34.42 capability_sasl_ir Option

The capability_sasl_ir IMAP option, when set to 1 (the default), causes the IMAP server to enable the SASL-IR IMAP extension. (See RFC 4959 (IMAP Extension for Simple Authentication and Security Layer (SASL) Initial Client Response).)

34.43 capability_searchres Option

The capability_searchres IMAP option, when set to 1 (the default), causes the IMAP server to enable the SEARCHRES IMAP extension. (See RFC 5182 (IMAP Extension for Referencing the Last SEARCH Result).)

34.44 capability_sort Option

The capability_sort IMAP option, when set to 1 (the default), causes the IMAP server to enable the SORT IMAP extension (RFC 5256 (Internet Message Access Protocol - SORT and THREAD Extensions)).

34.45 capability_sort_display Option

The capability_sort_display IMAP option, when set to 1 (the default), causes the IMAP server to enable the SORT=DISPLAY IMAP extension (RFC 5957).

34.46 capability_special_use Option

The capability_special_use IMAP option, when set to 1 (the default), causes the IMAP server to enable 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).

34.47 capability_starttls Option Under imap

The capability_starttls IMAP option, when set to 1 (the default), causes the IMAP server to enable the STARTTLS IMAP extension. (See RFC 2595 (Using TLS with IMAP, POP3, and ACAP) and RFC 3501 (IMAP4rev1).)

34.48 capability_status_size Option

The capability_status_size IMAP option, when set to 1 (the default), allows an IMAP client to ask the IMAP server for the size of all messages in a mailbox. For more information, see RFC 8438 IMAP Extension for STATUS=SIZE.

34.49 capability_thread_references Option

The capability_thread_references IMAP option, when set to 1 (the default), causes the IMAP server to enable the THREAD=REFERENCES IMAP extension. (See RFC 5256 (Internet Message Acces Protocol - SORT and THREAD Extensions).)

34.50 capability_thread_subject Option

The capability_thread_subject IMAP option, when set to 1 (the default), causes the IMAP server to enable the THREAD=ORDEREDSUBJECT IMAP extension. (See RFC 5256 (Internet Message Acces Protocol - SORT and THREAD Extensions).)

34.51 capability_uidplus Option

The capability_uidplus IMAP option, when set to 1 (the default), causes the IMAP server to enable the UIDPLUS IMAP extension. (See RFC 2359 (IMAP4 UIDPLUS extension).)

34.52 capability_unselect Option

The capability_unselect IMAP option, when set to 1 (the default), causes the IMAP server to enable the UNSELECT IMAP extension. (See RFC 3691 (Internet Message Access Protocol (IMAP) UNSELECT command).)

34.53 capability_url_partial Option

The capability_url_partial IMAP option, when set to 1 (the default), tells the IMAP client that it can use partial references in an IMAP URL. See RFC 5550 section 5.7 for more information.

34.54 capability_urlauth Option

The capability_urlauth IMAP option, when set to 1 (the default), causes the IMAP server to enable the URLAUTH IMAP extension. (See RFC 4467 (Internet Message Access Protocol (IMAP) - URLAUTH Extension).)

34.55 capability_utf8_accept Option

The capability_utf8_accept IMAP option, when set to 1 (the default), allows an IMAP client to enable use of UTF-8 mailbox names, quoted strings, and messages. See RFC 6855 IMAP Support for UTF-8 for more details.

Note that whether or not this is explicitly enabled by a client, the message store is presently liberal about allowing UTF-8 in the protocol and email messages with UTF-8 headers into the message store. The primary behavior difference when a client ENABLEs this extension is to disallow use of the modified-UTF-7 mailbox naming convention in the IMAP session and convert all mailboxes to UTF-8 when performing IMAP LIST operations.

34.56 capability_within Option

The capability_within IMAP option, when set to 1 (the default), causes the IMAP server to enable the WITHIN IMAP extension. (See RFC 5032 (WITHIN Search Extension to the IMAP Protocol).)

34.57 capability_x_netscape Option

The capability_x_netscape IMAP option, if set to 1, (the default being 0), causes the IMAP server to enable the X-NETSCAPE IMAP extension.

34.58 capability_x_orcl_as Option

The capability_x_orcl_as IMAP option, when set to 1 (the default), causes the IMAP server to enable 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.

34.59 capability_x_sun_imap Option

The capability_x_sun_imap IMAP option, when set to 1 (the default), causes the IMAP server to enable the X-SUN-IMAP IMAP extension.

34.60 capability_x_sun_sort Option

The capability_x_sun_sort IMAP option, when set to 1 (the default), causes the IMAP server to enable the X-SUN-SORT IMAP extension.

34.61 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. Starting with Messaging Server 8.1,

this has been replaced by the imap.capability_unauthenticate option. The default is 0. This should only be turned on when imap.capability_unauthenticate has been set to 1 and there is a need for backwards compatibility with a client that only recognizes X-UNAUTHENTICATE.

34.62 capability_unauthenticate Option

The capability_unauthenticate IMAP option, when set to 1, causes the IMAP server to advertise the UNAUTHENTICATE IMAP extension from RFC 8437. This is off by default (0).

34.63 capability_xrefresh Option

The capability_xrefresh IMAP option, when set to 1 (the default), causes the IMAP server to Enable the XREFRESH IMAP extension.

34.64 capability_xsender Option

The capability_xsender IMAP option, when set to 1 (the default), causes the IMAP server to advertise the XSENDER IMAP extension. In version 8.0 and later, the XSENDER feature was removed from imapd and this option has been deleted. DELETED: simplify message store by removing non-standard feature.

Important note: If the MMP imapproxy.capability option is explicitly configured, it is critical that XSENDER not be included in the value for MS 8.0 and later, as this can lead to clients trying to use the capability and failing. Similar considerations apply to third-party IMAP proxies that advertise a fixed set of capabilities.

34.65 capability_xserverinfo Option

The capability_xserverinfo IMAP option, if set to 1 (the default being 0), causes the IMAP server to enable the XSERVERINFO IMAP extension.

34.66 capability_xsnippet Option

The capability_xsnippet IMAP option, when set to 1 (the default), causes the IMAP server to enable the XSNIPPET IMAP extension (draft). This extension is experimental and subject to change in the event equivalent functionality is standardized.

34.67 capability_xum1 Option

The capability_xum1 IMAP option, when set to 1 (the default), causes the IMAP server to enable the XUM1 IMAP extension.

34.68 connlimits Option

The connlimits option (available under http, imap, pop, mmp, imapproxy, popproxy, 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 34.1 `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".

The option has no default value; however, initial configuration normally sets a value of `:20` for the IMAP Proxy, and POP Proxy:

```
msconfig> show connlimits
role.imapproxy.connlimits = :20
role.popproxy.connlimits = :20
```

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.

34.68.1 Use with isc_client

The `connlimits` isc_client option specifies the maximum number of connections that are permitted from a single LMTP process to the ISC server. Starting with 8.0.2.2, this isc_client option is deprecated; the `isc_client.max_conns` option should be used instead.

34.69 diacritical_sensitive_languages Option

By default, IMAP search is diacritical insensitive (meaning diacritical marks are ignored when searching for a string) while ISS search (as is used when imap.indexer.enable and imap.indexer.server_host are set) is diacriticial sensitive (meaning only exact diacritical matches are returned). This option can be used to make IMAP search diacritical sensitive for specific languages. The value is a space separated list of language tags (RFC 5646). If the IMAP LANGUAGE extension (RFC 5255) is used to negotiate a language in this list or a user's preferredLanguage LDAP attribute includes one of the languages in this list, then IMAP searches that are not sent to ISS will be diacritical sensitive instead of the default diacritical insensitive. If the language 'i-default' is included in this option then IMAP searches will be diacritical sensitive if the LANGUAGE extension is not used or the default language is explicitly selected.

34.70 domainallowed Option Under imap

The domainallowed IMAP option specifies access filters specifying which domains and/or IP addresses are allowed IMAP access.

34.71 domainnotallowed Option Under imap

The domainnotallowed IMAP option specifies access filters specifying which domains and/or IP addresses are not allowed IMAP access.

34.72 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 IMAP 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 IMAP option.

34.73 enableuserlist Option Under imap

The enableuserlist IMAP option enables imsconnutil connected user listing for IMAP service.

34.74 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.

34.75 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. With the default value of 1, this will read the date from the most recent received header in the message content and use that as the default internal date. When this is 0, the server's current time will be used as the default internal date. Set this to 0 if strict compliance with the IMAP standard is required. The default setting of 1 is more likely to behave as end-users expect when an IMAP client moves mail to the IMAP server.

34.76 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.

34.77 idletimeout Option Under imap

The idletimeout IMAP option specifies a maximum idle time, in minutes, for IMAP connections.

If an IMAP client has no activity for longer than the idletimeout value, then the IMAP server will close the connection with a message:

```
* BYE timeout surpassed
```

The default is 30 minutes, which RFC 3501 (IMAP v4rev1) specifies as the minimum such timeout. Attempting to set this option to a smaller value will result in a value of 30 being used (and an nslog Warning level message); attempting to set this option to a value larger than two days (larger than 2880=48*60) will result in a value of 48*60 being used (with no nslog message).

If using the MMP, see also its mmp.timeout option, where client IMAP connections can be affected by the setting under mmp as well as imapproxy.

34.78 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.

34.79 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).

34.80 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.

34.81 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.

34.82 logprotocolerrors Option Under imap

If the logprotocolerrors IMAP option is greater than zero, protocol errors are logged as debug messages for IMAP.

34.83 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.

34.84 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. A legacy configutil option, service.imap.maxmessagesize is introduced in 8.0.2.1.

34.85 maxnoops Option

The maxnoops IMAP option specifies the maximum number of NOOP commands accepted before connection is forcibly closed.

34.86 maxprotocolerrors Option Under imap

The maxprotocolerrors IMAP option specifies the maximum number of protocol errors allowed before the IMAP connection is forcibly closed.

34.87 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.

34.88 maxsearchnest Option

The maxsearchnest IMAP option specifies the maximum nesting depth in an IMAP SEARCH command. If this limit is exceeded, the search command will return an error. Nesting of at least 20 levels is always permitted regardless of this setting.

34.89 maxsessions Option Under imap

The maxsessions IMAP option specifies the maximum number of sessions per IMAP server process.

34.90 maxthreads Option Under imap

The maxthreads IMAP option specifies the maximum number of threads per IMAP server process.

34.91 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.

34.92 plaintextmincipher Option Under imap

If the imap.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.

34.93 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 noticably 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.

34.94 port Option Under imap

The port IMAP option specifies the IMAP server port number. The default is 143.

34.95 sslcachesize Option Under imap

The sslcachesize IMAP option specifies the number of SSL sessions to be cached by the IMAP server. If this is set to 0 or not set, this will use a default provided by the Mozilla NSS library which was 10000 last time this was checked (March 2016).

34.96 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).

34.97 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 IMAP option must be set also. (In MS 7.0.5, it was also necessary to set the sslusessl IMAP option *explicitly* to 1 even though the default was 1; but as of MS 8.0, setting sslusessl is not necessary for the IMAP server.)

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.

34.98 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 imap.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.

34.99 submituser Option

The submituser IMAP option specifies the Message Store userid used by the MTA when resolving submit IMAP URLs in BURL commands.

34.100 withinresolution Option

The withinresolution IMAP option specifies the interval (in minutes) between recalculations of Contexts involving the search options YOUNGER or OLDER.

34.101 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.

See also the pwchangeurl Base option.

34.101.1 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.

34.101.2 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 is 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.

34.101.3 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 35 POP options

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 `logfile` options, the various `bg*` options, and `defaultdomain`.

Of notable relevance to POP operation, see also the `base.threadholddelay` and `base.dnsresolveclient` options.

Note that msprobe can probe for whether the POP server is running; see msprobe's `msprobe.probe:pop.` options.

The MSHTTP server can optionally be configured to perform collection of messages from remote POP servers; see the allowcollect, maxcollectmsglen, and popbindaddr MSHTTP options.

35.1 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.enablesslport in Unified Configuration, or service.pop.enablesslport in legacy configuration.

This option defaults to 0 if not set, but initial configuration may enable the option as appropriate.

35.2 actions Option

The actions option (available for imap, pop, and messagetrace) specifies the actions enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(a1 a2 a3)" to enable only attributes listed, or "-(a1 a2 a3)" to enable all attributes except those listed. See the Store Transaction Log Format section for permitted attribute codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(fe)" (log everything except the fetch action) by default.

35.3 actionattributes Option

The actionattributes option (available for imap, pop, and messagetrace) specifies the action attributes enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(e1 e2 e3)" to enable only event codes listed, or "-(e1 e2 e3)" to enable all events except those listed. See the Store Transaction Log Format section for permitted event codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(mi)" (log all attributes except the message id attribute) by default. Note that the message id is expensive to extract from message headers and requires per-message logging of expunge events.

35.4 allowanonymouslogin Option Under pop

The allowanonymouslogin POP option sets whether or not anonymous login is allowed by POP.

35.5 authfaildelay Option

The authfaildelay option (available under IMAP and POP) determines how long the server delays before reporting an authentication failure. This option is present only for the back-end POP3 and IMAP servers. The MMP uses a more sophisticated badguyslist facility (see bgpenalty). Decreasing the authfaildelay option below the default value (3) is not recommended.

35.6 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).

35.7 bgmax Option

The bgmax option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See bgpenalty for more information.

35.8 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.

The bgpenalty option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the length of time in seconds added to the authentication delay after each failed authentication.

35.9 bgmaxbadness Option

The bgmaxbadness option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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.

35.10 bgdecay Option

The bgdecay option (available under base, imap, pop, mmp, imapproxy, and popproxy) represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See bgpenalty for more information.

35.11 bglinear Option

The bglinear option (available under base, imap, pop, mmp, imapproxy, and popproxy) defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See bgpenalty for more information.

35.12 bgexcluded Option

The bgexcluded option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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).

35.13 connlimits Option

The connlimits option (available under http, imap, pop, mmp, imapproxy, popproxy, 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 35.1 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".

The option has no default value; however, initial configuration normally sets a value of :20 for the IMAP Proxy, and POP Proxy:

```
msconfig> show connlimits
role.imapproxy.connlimits = :20
role.popproxy.connlimits = :20
```

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.

35.13.1 connlimits Option Under isc_client

The connlimits isc_client option specifies the maximum number of connections that are permitted from a single LMTP process to the ISC server. Starting with 8.0.2.2, this isc_client option is deprecated; the isc_client.max_conns option should be used instead.

35.14 domainallowed Option Under pop

The domainallowed POP option specifies access filters specifying which domains and/or IP addresses are allowed POP access.

35.15 domainnotallowed Option Under pop

The domainnotallowed POP option specifies access filters specifying which domains and/or IP addresses are not allowed POP access.

35.16 emulateqpopper Option

Hack.

35.17 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 POP 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 POP option.

35.18 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.

35.19 idletimeout Option Under pop

The idletimeout POP option specifies a maximum idle time, in minutes, for POP connections. After this timeout, an idle POP connection will be disconnected. The default is 10 minutes. Only values between 10 and 1440 (24*60) will be used; attempting to set a value outside this range will result in the nearest permissible value being used.

If using the MMP, see also its mmp.timeout option, where client POP connections can be affected by the setting under mmp as well as popproxy.

35.20 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.

35.21 logprotocolerrors Option Under pop

If the logprotocolerrors POP option is greater than zero, protocol errors are logged as debug messages for POP.

35.22 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.

35.23 maxprotocolerrors Option Under pop

The maxprotocolerrors POP option specifies the maximum number of protocol errors allowed before the POP connection is forcibly closed.

35.24 maxsessions Option Under pop

The maxsessions POP option specifies the maximum number of sessions per server process.

35.25 maxthreads Option Under pop

The maxthreads POP option specifies the maximum number of threads per POP server process.

35.26 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.

35.27 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.

35.28 poplogmboxstat Option

The poplogmboxstat POP option, if set to 1 (the default being 0), causes the POP log to show mailbox statistics on login and logout.

35.29 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.

35.30 port Option Under pop

The port POP option specifies the POP server port number. The default is 110.

35.31 sslcachesize Option Under pop

The sslcachesize POP option specifies the number of SSL sessions to be cached by the POP server. If this is set to 0 or not set, this will use a default provided by the Mozilla NSS library which was 10000 last time this was checked (March 2016).

35.32 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).

35.33 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 POP option must also be set. (In MS 7.0.5 it was also necessary to set the sslusessl POP option *explicitly* to 1 even though the default was 1; but as of MS 8.0, the POP server does not require that option.)

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.

35.34 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 pop.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.

Chapter 36 Message Trace options

Message Trace can be enabled by setting the `messagetrace.activate` option to `transactlog` (recommended), `yes` (deprecated), or `messagetrace` (deprecated). 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 `transactlog.logfile.*` (for `transactlog` log file) or `messagetrace.logfile.*` (for `messagetrace` log file).

36.1 `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 `no`, `yes`, `msgtrace`, and `transactlog`. Specifying `msgtrace` causes message tracing to be written in traditional format, to a file named `msgtrace`. Specifying `transactlog` causes message trace entries to be written in an easy-to-parse XML format, similar to the MTA's XML format message transaction entries, to a file named `transactlog`. As of MS 8.0.1, the Message Store `msgtrace` log format (selected via `msgtrace`) is deprecated in favor of the Message Store action log format (selected via `transactlog`) which uses an easy-to-parse format (XML) described in the Store Transaction Log Format section.

Note that the default values for the actions and actionattributes have changed in 8.0.2; they both need to be set to "all" explicitly to get functionality that is a superset of the old `msgtrace` or `yes` settings.

Starting with the Messaging Server 8.1 release, the legacy `msgtrace` format has been removed so both `msgtrace` and `yes` values are treated as synonyms for `transactlog`.

36.2 `actions` Option

The `actions` option (available for `imap`, `pop`, and `messagetrace`) specifies the actions enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(a1 a2 a3)" to enable only attributes listed, or "-(a1 a2 a3)" to enable all attributes except those listed. See the Store Transaction Log Format section for permitted attribute codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(fe)" (log everything except the fetch action) by default.

36.3 `actionattributes` Option

The `actionattributes` option (available for `imap`, `pop`, and `messagetrace`) specifies the action attributes enabled in Message Store transaction logging. This can take the value of "all" to enable logging of all actions, or "+(e1 e2 e3)" to enable only event codes listed, or "-(e1 e2

e3)" to enable all events except those listed. See the Store Transaction Log Format section for permitted event codes.

For 8.0.1 this is "all" by default and for 8.0.2 this is "-(mi)" (log all attributes except the message id attribute) by default. Note that the message id is expensive to extract from message headers and requires per-message logging of expunge events.

36.4 loglevel Option Under messagetrace

Legacy 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. Data in the `transactlog` is inherently `critical` level.

Chapter 37 `notifytarget` options

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
```

37.1 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.

37.2 notifytype Option

The notifytype option under notifytarget specifies the type of this notification target. Presently the value can be either ens (the default) or jmq. Note that support for jmq 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.3 enseventkey Option Under notifytarget

The enseventkey option under notifytarget, notifytarget:*target-name*.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.

37.4 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.

37.5 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).

Prior to MS 8.0, this option defaulted to a value of 7997. As of MS 8.0, if ens.enablesslport (local.ens.enablesslport in legacy configuration) is set,

then this option defaults to the value of ens.sslport (local.ens.sslport in legacy configuration) if it is set or 8997 if the sslport is not explicitly set; otherwise, this option defaults to the value of ens.port (local.ens.port in legacy configuration) if it is set to a valid value, or 7997 if ens.port does not have a valid value.

37.6 enspwd Option

The enspwd option under notifytarget, notifytarget:*target-name*.enspwd, specifies the ENS Broker user password that is used to authenticate to the ENS broker. The default value of the option will be the value of the option ens.secret, if the *target-name* of the notifytarget is ms-internal or the notifytarget:*target-name*.enshost is not set or the notifytarget:*target-name*.enshost is same as the value of the listenaddr Base option (service.listenaddr in legacy configuration) or the notifytarget:*target-name*.enshost is 127.0.0.1 or the notifytarget:*target-name*.enshost is ::1.

37.7 ensuser Option

The ensuser option under notifytarget, notifytarget:*target-name*.ensuser, specifies the ENS username. The default is "guest".

37.8 ensusessl Option

The ensusessl option is available for notifytargets. The defaulting is special for notifytarget:ms-internal.

37.8.1 Use with notifytarget

The ensusessl option under notifytarget, notifytarget:*target-name*.ensusessl, specifies whether the connection to the ENS broker for this notification target should use TLS/SSL or not. This works only if the notifytype of the notifytarget is equal to "ens". Setting ensusessl to 1 directs the notifytarget to connects to the specified ENS enshost using TLS/SSL.

Arbitrary, remote targets of notifytarget default to an ensusessl value of 0. But if the *target-name* of the notifytarget is ms-internal, the default value will be equal to value of the enablesslport ENS option.

37.8.2 Use with ms-internal Under notifytarget

The ensusessl option under notifytarget:ms-internal, notifytarget:ms-internal.ensusessl, specifies whether the connection to the ENS broker for this notification target should use TLS/SSL or not. This works only if the notifytype of notifytarget:*target-name* is equal to "ens". Setting ensusessl to 1 directs the notifytarget to connect to the specified ENS enshost using TLS/SSL.

While arbitrary, remote targets of notifytarget default to 0, for ms-internal, the default value will be equal to value of the enablesslport ENS option.

37.9 jmqhost Option Under notifytarget

The jmqhost option under notifytarget, notifytarget:*target-name*.jmqhost, specifies the hostname of the JMQ broker.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.10 jmqport Option Under notifytarget

The jmqport option under notifytarget, notifytarget:*target-name*.jmqport, specifies the port number of the JMQ broker. The default is 7676.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.11 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. Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.12 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.13 jmquser Option Under notifytarget

The jmquser option under notifytarget, notifytarget:*target-name*.jmquser, specifies the JMQ username. The default is "guest".

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.14 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.15 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. If the body is longer than this, it will be truncated.

37.16 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. If the header is longer than this, it will be truncated and padded with a blank line.

37.17 msgflags Option Under notifytarget

The msgflags option under notifytarget, notifytarget:*target-name*.msgflags, enables the msgflag notification mechanism.

37.18 destinationtype Option

The destinationtype option under notifytarget, notifytarget:*target-name*.destinationtype, specifies the JMQ destination type, queue or topic (the default).

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.19 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.20 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.21 priority Option

The priority option under notifytarget, notifytarget:*target-name*.priority, specifies the priority to be used for JMQ notification messages.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.22 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.

Starting with the Messaging Server 8.1 release, the JMQ notifytargets feature is deprecated. This feature is only supported for customers migrating from ISS to Elasticsearch.

37.23 deletemsg Option Under notifytarget

The deletemsg option under notifytarget, notifytarget:*target-name*.deletemsg, specifies whether DeleteMsg events will generate a notification.

37.24 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.

37.25 newmsg Option Under notifytarget

The newmsg option under notifytarget, notifytarget:*target-name*.newmsg, specifies whether NewMsg events will generate a notification.

37.26 overquota Option Under notifytarget

The overquota option under notifytarget, notifytarget:*target-name*.overquota, specifies whether OverQuota events will generate a notification.

37.27 underquota Option Under notifytarget

The underquota option under notifytarget, notifytarget:*target-name*.underquota, specifies whether UnderQuota events will generate a notification.

37.28 setacl Option

The setacl option under notifytarget, notifytarget:*target-name*.setacl, specifies whether SetAcl events will generate a notification.

37.29 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.

37.30 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.

37.31 purgemsg Option Under notifytarget

The purgemsg option under notifytarget, notifytarget:*target-name*.purgemsg, specifies whether PurgeMsg events will generate a notification.

37.32 readmsg Option Under notifytarget

The readmsg option under notifytarget, notifytarget:*target-name*.readmsg, specifies whether ReadMsg events will generate a notification.

37.33 updatemsg Option Under notifytarget

The updatemsg option under notifytarget, notifytarget:*target-name*.updatemsg, specifies whether UpdateMsg events will generate a notification.

37.34 expungemsg Option

The expungemsg option under notifytarget, notifytarget:*target-name*.expungemsg, specifies whether ExpungeMsg events will generate a notification.

37.35 annotatemsg Option

The annotatemsg option under notifytarget, notifytarget:*target-name*.annotatemsg, specifies whether AnnotateMsg events will generate a notification.

37.36 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.

37.37 copymsg Option

The copymsg option under notifytarget, notifytarget:*target-name*.copymsg, specifies whether IMAP COPY operations will generate Copy events or UpdateMsg events.

Chapter 38 IMAP error statuses

When users attempt to access or manipulate their mailboxes or messages via an IMAP client or POP client may, various Message Store errors may occur and be reported or logged; these errors are referred to as *IMAP errors*, since the Message Store is a fundamentally IMAP-oriented repository of messages. Utilities operating on the Message Store may also encounter such IMAP errors. And MTA channels attempting to deliver to the Message Store, *i.e.*, `ims-ms` channels or `tcp_lmtpcs*` channels, can also encounter a subset of such IMAP errors (discussed further in ims-ms channel error messages).

Additional details on the underlying Message Store issue that gave rise to an IMAP error status may be available in the Message Store NSLOG logging for the component which encountered the issue; in the case of the `ims-ms` channel or LMTP server, see the `imta` NSLOG file discussed in ims-ms channel debugging and error logging.

Table 38.1 IMAP error statuses

Name	Type	RFC 5530 code	Text+	Meaning
IMAP_IOERROR	MTA-temp	[CONTACTADMIN]	System I/O error. Administrator, check server log for details.	The Message Store may be corrupted, or otherwise inaccessible (*e.g.*, disk not mounted); or the Message Store `store.idx` file may have reached its 2 gigabyte size limit.
IMAP_CONFIG_ERROR	MTA-perm	[CONTACTADMIN]	Configuration error	For the MTA Message Store delivery channels, returned if either LDAP pool code, or the domain map code, cannot be initialized. For Message Store servers and Message Store utilities, returned if the Message Store or library code cannot be initialized.
IMAP_PERMISSION_DENIED	MTA-perm	[NOPERM]	Permission denied	Typically, this error relates to an attempt to create or delete a mailbox (*i.e.*, folder), where either there was a (semantic) problem with the folder name, or a problem with IMAP ACLs on the folder. (Note that for privacy reasons, IMAP ACL problems may instead be reported with the less informative IMAP_MAILBOX_NONEXISTENT error.) This error may also be reported in cases of ACL problems attempting to read or modify an IMAP flag on a message.
IMAP_QUOTA_EXCEEDED	MTA-temp	[OVERQUOTA]	Over quota	The recipient has exceeded their configured quota, but for less than the configured grace period.
IMAP_QUOTA_EXCEEDED_PERSISTENT	MTA-perm	[OVERQUOTA]	Over quota	The user has exceeded their configured quota for more than the configured grace period.
IMAP_MESSAGE_TOO_LARGE	MTA-perm	[LIMIT]	Message too large	Message is larger than the user's entire quota.
IMAP_USERFLAG_EXHAUSTED		[LIMIT]	Too many keywords in mailbox	
IMAP_MAILBOX_BADFORMAT	MTA-temp	[CORRUPTION]	Mailbox has an invalid format	The mailbox may be corrupted.
IMAP_MAILBOX_NOTSUPPORTED	MTA-temp		Operation is not supported on mailbox	The mailbox may be corrupted.
IMAP_MAILBOX_NONEXISTENT	MTA-perm	[NONEXISTENT]	Mailbox does not exist	(Recall that these error messages are general IMAP error messages, that may occur in other contexts besides `ims-ms` channel or LMTP channel delivery.) When this error occurs in the context of an `ims-ms` or LMTP channel delivery, the meaning is as follows. Normally, the `ims-ms` channel and LMTP server `tcp_lmtpss*` channel will create a mailbox, if it does not exist. And when delivery to a specific folder is being attempted due to a Sieve `"fileinto"` action, and the folder does not exist, the channel will attempt to create the folder. (If the folder part of the address was not generated by the user's Sieve filter nor world writeable, then the channel will log a General facility, Information level notice to the `imta` file saying "`append_setup` *`path`* `failed, trying INBOX:`" followed by the IMAP_MAILBOX_NONEXISTENT error text. If the folder part was due to the user's Sieve filter but the folder could not be created, then the channel will log a General facility, Error level notice to the `imta` file saying "`mboxlist_createmailbox` *`path`* `failed, trying INBOX:`" with the specific IMAP error text.) But if the partition is bogus, then this error will be returned.
IMAP_WRONG_MAILHOST	MTA-perm		Mailbox is on a different server	Arises when attempting to create a mailbox (folder). For the MTA's delivery channels, this suggests that either

				the user's LDAP entry is missing a `mailHost` attribute entirely, or their `mailHost` has been changed (their mailbox has been moved) while old messages were awaiting delivery in the `ims-ms` channel queue area, or that there is an error in the MTA configuration (causing it to attempt to deliver a message on the wrong `mailHost`), or operator error whereby messages have been manually moved (incorrectly) from one host to another.
IMAP_MAILBOX_EXISTS		[ALREADYEXISTS]	Mailbox already exists	When attempting to create a folder, this is a success status
IMAP_MAILBOX_BADNAME	MTA-perm	[CANNOT]	Invalid mailbox name	
	Delivery temporary errors			
IMAP_MAILBOX_LOCKED	MTA-temp	[INUSE]	Mailbox is busy	The mailbox was locked. When this status is encountered, the MTA delivery channel (`ims-ms` or `tcp_lmtpcs*`) first retries 10 times at 100 millisecond intervals (before this error is ever even reported back from the channel). Prior to 8.0, lock attempts were nonblocking; a 1 second time is used in 8.0 or later. If the mailbox remains locked, then the channel gives up on this delivery attempt, generating a "`Q`" record with the "`Mailbox is busy`" reason if `logging` is enabled. And then there is special code in the `ims-ms` channel (and as of MS 7.0 in the LMTP client channel; see the MAILBOX_BUSY_FAST_RETRY TCP/IP-channel-specific option) to ask the Job Controller to retry delivery again very, very soon---with a randomly generated backoff of between one second and two minutes---overriding the normal `*backoff` values. Occasional occurrences of such a temporary error are normal, as for instance if a user is logged in and performing extensive, time-consuming operations on a mailbox; indeed, it is because an occasional such occurrence is "normal" that the delivery channels have special code to handle this (usually transient) error condition by retrying delivery very soon. However, persistent occurrences for the same user might suggest a problem with that user's mailbox. Or, if such errors suddenly start occurring at around the same time for all (or many) users, that could be a suggestion of some more widespread and general problem with the Message Store.
IMAP_MAILBOXLIST_NONEXISTENT		[NONEXISTENT]	Mailbox list does not exist	
IMAP_MAILBOX_EXHAUSTED		[LIMIT]	Too many mailboxes	IMAP client attempt to perform an IMAP search on more than `maxsearchmailboxes`.
IMAP_MAILBOX_PINNED		[NOPERM]	Mailbox is pinned	Attempt to deleted a "pinned" mailbox (that is, folder); see the `pin` Message Store option.
IMAP_PARTITION_UNKNOWN	Temporary		Unknown/invalid partition	The user's `mailMessageStore` LDAP attribute does not correspond to a *`partition-name`* defined via a `path` partition option in Unified Configuration (a `store.partition.`*`partition-name`*`.path` configutil parameter in legacy configuration), or the value of that option or configutil parameter does not point to a valid location, or contains invalid characters (only alphanumeric characters are allowed). (If a user does not have a `mailMessageStore` set at all, then the Message Store's default partition, `defaultpartition` Message Store option in Unified Configuration or `store.defaultpartition` configutil parameter in legacy configuration, is assumed, which defaults to `primary`. When the absence of an explicit `mailMessageStore` is causing use of the default partition, then whatever the default partition is, it must then have a valid path specified in the `path` option for that named partition in Unified Configuration (`partititon:`*`whatever`*`.path`) or the `store.partition.`*`whatever`*`.path` configutil parameter; in particular, when `store.defaultpartition` has its default value of `primary`, then the primary partition must have a valid path specified in `partition:primary.path` in Unified Configuration or `store.partition.primary.path` in legacy configuration.)
IMAP_INVALID_IDENTIFIER		[CANNOT]	Invalid identifier	May occur when attempting to set an ACL on a folder
IMAP_INVALID_MSGNO			Invalid message number	
IMAP_MESSAGE_CONTAINSNULL	MTA-perm	[UNKNOWN-CTE]	Message contains NUL characters	(A message that has gone through the MTA will not normally cause such an error, as the MTA will normally legalize message content.)
IMAP_MESSAGE_BADHEADER	MTA-perm		Message contains invalid header	
IMAP_MESSAGE_NOBLANKLINE	MTA-perm		Message has no header/body separator	(A message that has gone through the MTA will not normally cause such an error, as the MTA will

				normally legalize messages one way or another; see the `*headertermination` channel options.)
IMAP_MESSAGE_EXHAUSTED		[LIMIT]	Too many messages	
IMAP_QUOTAROOT_NONEXISTENT			Quota root does not exist	
IMAP_MAILBOX_TOO_LARGE		[LIMIT]	Mailbox too large	
IMAP_ANNOTATEMORE_TOOBIG		[LIMIT]	Data too big	
IMAP_UIDVALIDITY_INCORRECT			Mailbox has been replaced by a newer version	
IMAP_UNRECOGNIZED_CHARSET		[BADCHARSET]	Unrecognized character set	
IMAP_UNRECOGNIZED_LANGUAGE			Unrecognized language	
IMAP_INVALID_USER	MTA-perm		Invalid user	The recipient address did not parse into syntactically valid uid, channel part, and optionally domain and/or folder portions
IMAP_INVALID_LOGIN		[AUTHENTICATIONFAILED]	Login incorrect	
IMAP_ANONYMOUS_NOT_PERMITTED	IMAP-only		Anonymous login is not permitted	Returned by the IMAP server to any attempt to login as "`anonymous`" when `imap.allowanonymouslogin` has *not* been set.
IMAP_USER_NOT_PERMITTED		[AUTHORIZATIONFAILED]	User not permitted	
IMAP_TEMP_AUTH_FAILURE		[UNAVAILABLE]	Authentication server temporarily unavailable	
IMAP_UNSUPPORTED_QUOTA			Unsupported quota resource	
IMAP_UNRECOGNIZED_COMPARATOR		[BADCOMPARATOR]	Unrecognized comparator	
IMAP_BADURLPART			nonexistant message section	
IMAP_BADURL			Invalid or inappropriate URL	
IMAP_JUNK_AT_END			Junk at end of command	
IMAP_NO_OVERQUOTA		[OVERQUOTA]	Mailbox is over quota	
IMAP_NO_CLOSEQUOTA		[ALERT]	Mailbox is at <x>% of quota	
IMAP_NO_MSGGONE			Message no longer exists	
IMAP_NO_MSGTMPFAIL		[UNAVAILABLE]	Message temporarily unavailable	
IMAP_NO_DECODE		[UNKNOWN-CTE]	Unable to decode message	
IMAP_NO_CHECKPERUSER			Unable to checkpoint per-user flags	
IMAP_NO_CHECKPRESERVE			Unable to preserve per-user flags	
IMAP_PROXY_UNAVAILABLE		[UNAVAILABLE]	Server hosting this mailbox is not available	
IMAP_PROXY_ERROR		[CONTACTADMIN]	Error parsing backend IMAP server response	
IMAP_PROXY_DISCONNECTED		[UNAVAILABLE]	The backend IMAP server has disconnected	
IMAP_PROXY_CONDSTORE_ERROR			The backend IMAP server does not support CONDSTORE	
IMAP_PARTITION_FULL	MTA-temp	[CONTACTADMIN]	Store partition is full	See also the (new in MS 6.2) configutil parameters `local.store.checkdiskusage` and `local.store.diskusagethreshold` (initially instead named `local.store.diskthreshold` in MS 6.2), or in Unified Configuration the `checkdiskusage` and `diskusagethreshold` Message Store options, which control whether---and when---the store performs checks on its disk space usage. With such checks enabled, the store will "lock" a partition (at which point the IMAP_PARTITION_FULL error will be returned) slightly before the partition is actually filled up. This is a safety measure, to ensure that expunge operations (which need to rewrite the index files) have disk room in which to operate.

IMAP_PROXY_ONLY	MTA-perm		Server is not configured for local users	
IMAP_REPLACEMENT_STRING_TOO_LONG			unknown-character-replacement value is too long	
IMAP_CANNOT_CONVERT_FROM_CHARSET			Charset of original body unrecognised	
IMAP_CANNOT_CONVERT_TO_CHARSET			Destination charset unrecognized	
IMAP_CONVERT_ERROR			Conversion failed	
IMAP_BAD_CONVERT_PARAMETERS			Bad convert parameter specified	
IMAP_MISSING_CONVERT_PARAMETER			Required convert parameter missing	
IMAP_UNKNOWN_CHARACTER_FOUND			unknown character found. try specifying unknown-character-replacement	
IMAP_NO_CONVERTER			No converter defined for this pair of body types	
IMAP_CONVERT_MULTIPART			Cannot convert whole message or multipart parts	
IMAP_URLERR_SCHEME			Invalid URL scheme found in URL	
IMAP_URLERR_RELPATH			Relative path found in URL	
IMAP_URLERR_USER			Invalid user name found in URL	
IMAP_URLERR_PASS			Invalid password found in URL	
IMAP_URLERR_HOST			Invalid host found in URL	
IMAP_URLERR_PORT			Invalid port found in URL	
IMAP_INVALID_ACCESS		[NOPERM]	Access denied	
IMAP_ANNOTATE_TOOBIG		[ANNOTATE TOOBIG]	Annotation too large.	
IMAP_ANNOTATE_TOOMANY		[ANNOTATE TOOMANY]	Too many annotations.	
IMAP_INVALID_USEATTR		[USEATTR]	Use attribute not supported.	
IMAP_MESSAGE_CONTAINSNL	Permanent		Message contains bare newlines	(A message that has gone through the MTA will not normally cause such an error, as the MTA will normally canonicalize message content resulting in proper use of CRLF for line breaks, and encoded newlines in binary content; see the discussion of the `lmtp*` and `smtp*` channel options.)
Untagged BYE responses				
IMAP_BYE_LOGOUT			LOGOUT received	
IMAP_BYE_SHUTDOWN			System shutting down	
IMAP_ACCOUNT_TEMP_UNAVAILABLE		[UNAVAILABLE]	Account temporarily unavailable for system maintenance	
Tagged OK responses				
IMAP_OK_COMPLETED			Completed	
Password expiration warnings				
IMAP_PWEXPIRE_LESS1DAY		[ALERT]	Your password will expire in less than one day.	See the IMAP password expiration alert options.
IMAP_PWEXPIRE_1DAY		[ALERT]	Your password will expire in one day.	See the IMAP password expiration alert options.
IMAP_PWEXPIRE_NDAYS		[ALERT]	Your password will expire in <n> days.	See the IMAP password expiration alert options.
IMAP_PWCHANGE_URL			You may change your password by directing your web browser to <url>.	Additional text added to above password expiration warnings, if the pwchangeurl base option is set (or if it is not set, if the `accounturl` base option is set).
Used when archiving causes APPEND to fail				

IMAP_ARCHIVE_FAILAPPEND		[CONTACTADMIN]	Mailbox append disabled due to archiving failure.	If the `compliance` Message Store archive option is set, but the archiving could not be performed during an attempted IMAP APPEND operation, this is the error returned.
Additional IMAP errors				
IMAP_FSCORRUPT			File system error. Administrator, check server log for details.	Typically means that the cache file or index file is corrupted.
IMAP_SYS_ERROR			System error. Administrator, check server log for details (new in 8.0.2).	
IMAP_SERVER_TEMP_BUSY			Server temporary busy. Administrator, check server log for details (new in 8.0.2).	
IMAP_REFER_USER			Follow referral to access INBOX and personal folders.	IMAP server referral attempt.
IMAP_REFER_MASTER			This is a read-only replica, follow referral for read-write master.	IMAP server referral attempt.
IMAP reporting underlying authentication (HULA) errors				
HULA_TRANS		[TRANSITION-NEEDED]	A transition is needed to use the specified mechanism	
HULA_EXPIRED		[EXPIRED]	Password expired	
HULA_NOMEM		[UNAVAILABLE]	Not enough memory available	
HULA_TRYAGAIN		[UNAVAILABLE]	Transient failure -- try authentication again	
HULA_UNAVAIL		[UNAVAILABLE]	Service temporarily unavailable	
HULA_NOAUTHZ		[AUTHORIZATIONFAILED]	Not authorized to login as specified user	
HULA_DISABLED		[AUTHORIZATIONFAILED]	Account disabled	Domain status, or user status, is "`disabled`"
HULA_ENCRYPT		[PRIVACYREQUIRED]	Encryption required	
HULA_*all-other-errors*		[AUTHENTICATIONFAILED]	*error-detail-text*	All other authentication (HULA) errors.

[+] The IMAP error text is localizable. The text shown in this table is merely the default text (which happens to be English).

Chapter 39 User identifiers

A user in a mail store can have multiple user identifiers for different purposes; this section provides a summary of some of the possible identifiers for a user:

Canonical Identifier

The canonical identifier for a user is typically derived from an LDAP lookup. In a default configuration, this is the value of the LDAP entry's uid followed by "@" and the domain name containing the user. The ldap_permid option (or if that's not set, the ldap_uid option) determine which LDAP attribute is used to construct this identifier. The `domainUidSeparator` domain LDAP attribute also alters how this attribute is constructed. This is also the value used for `%s` in the canonicalsearchfilter option.

External Identifier

For Cassandra store, this is the identifier used in IMAP ACL commands and when referencing shared folders through IMAP. Both ACL identifiers and shared folder names are internally stored using the store user identifer, but that is converted to and from the external identifier (if one exists) when IMAP is used. The ldap_extid option must be set to specify the LDAP attribute containing the external identifier, otherwise the store user identifier will be used.

LDAP Distinguished Name (DN)

The LDAP distinguished name provides a unique reference to a user entry in LDAP. Messaging Server does not have any specific requirements on which attributes are present in a user LDAP DN, although it is recommended that a subtree is created in LDAP for each set of users associated with one or more domain names. Messaging Server does not support multi-valued RDNs.

Original Login Identifier

The original login identifier is the identifier the user sends over the wire when performing a login or authentication operation. It may or may not be qualified by a domain name. There can be multiple valid login identifiers for the same user, but a given login identifier should uniquely match an LDAP entry. For LDAP search filter template options such as searchformat and replayformat, this identifier is referred to with the %o substitution. The `inetDomainSearchFilter` LDAP domain attribute typically determines how this is translated into a specific user's LDAP entry.

Permanent or Persistent User Identifier

Another name for the store user identifier, but explicitly noting that the identifier should be permanent and/or persistent. The ldap_permid option controls the LDAP attribute that contains this identifier for a given user.

Pre-Lookup Login Identifier

The pre-lookup login identifier is derived from the original login identifier by making sure it is qualified by the appropriate domain name. This includes converting any non-default domain delimiter (specified by the

loginseparator option) into the canonical delimiter (typically '@' or the first character listed in that option). This is determined prior to performing LDAP lookups for authentication purposes.

Store User Identifier

The store user identifier is the same as the canonical identifier, except that if the domain name is the default domain, then the domain portion of the identifier is omitted. Message store command line tools use this identifier when referring to users. For the classic message store, mboxutil can be used to migrate the content of a store user identifier's account to a new user identifier (due to IMAP ACLs and shared folder subscriptions, the process is not fast and will cause problems when the user being renamed is logged in). For Cassandra store, this identifier can not be changed; instead an external identifier should be used as that can be changed easily.

Part V Proxies and the MMP

Proxies and the MMP, for accessing the Message Store have a number of configuration options.

Chapter 40 Proxy options

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.

40.1 `httpadmin` Option

DEPRECATED: See the `proxyadmin` Base option instead.

Store admin login name for a specific host if different from `local.service.http.proxy.admin`. Not configured by default.

40.2 `httpadminpass` Option

DEPRECATED: See the `proxyadminpass` Base option instead.

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.

40.3 `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.

40.4 `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.

40.5 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).

40.6 storehostlist Option

The storehostlist Proxy option specifies a list of server hostnames that have access to the same message store data. For classic store, 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. For Cassandra store, the hosts are considered equivalent but connections are attempted in order.

When a mailHost attribute is found in LDAP, it is first resolved via this configuration. This allows MMP, mshttpd, imapd and MTA clients (with affinitylist) to try connecting to the mail store via an alternative host. The hostnames used in this option must match the hostnames in the mailHost attributes exactly (although ASCII case variations are permitted).

The value of the base.hostname option must be present in the storehostlist if you want the current host to participate in that replication group.

40.7 hostselect Option

The hostselect Proxy option specifies how the back-end server is selected from a list of possible hosts in the storehostlist option when a front-end service such as the MMP, mshttpd or LMTP client connects to a back-end service. The failover value routes all connections to the first host in the list or the host that was most recently successful for this affinity group. This value is recommended when using the automatic failover feature of the classic message store. The roundrobin value uses each host in the list in sequence to provide limited load balancing between the hosts and is recommended for use with the Cassandra message store.

Chapter 41 MMP and IMAP Proxy and POP Proxy and vdomain options

There are many options affecting MMP operation, or operation of its subcomponents the IMAP Proxy, or the POP 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 41.1 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`	✓ *complex*	✓ *complex*	✓ *complex*		
`bethegroup`: 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`		✓ *very long list---see text*			
`certmapdn`: alias for `usergroupdn`					
`certmapfile`: deleted					
`connecttimeout`	✓ 30	✓ 30	✓ 30		
`connlimits`	✓ :20	✓	✓	✓	
`connrejectthreshold`	✓ *complex---see text*				
`crams`	✓ 0	✓ 0	✓ 0		✓ 0
`debugkeys`	✓	✓	✓	✓	✓
`defaultdomain`	✓	✓	✓	✓	✓
`domainallowed`		✓	✓	✓	
`domainnotallowed`		✓	✓	✓	
`domainsearchformat`	✓ `(uid=%U)`	✓ `(uid=%U)`	✓ `(uid=%U)`		✓ `(uid=%U)`
`enable`	✓ 0				
`hosteddomains`	✓ 1	✓ 1	✓ 1		✓ 1
`ipv6in`	✓ 0				
`ipv6out`	✓ 0				
`ipv6sortorder`	✓ `default`				
`langlist`	✓ `i-default EN`	✓ `i-default EN`			
`ldapcachesize`	✓ *complex*	✓ *complex*	✓ *complex*	✓ *complex*	✓ *complex*

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	✓	✓	✓	✓	
logfile.*option-name*	✓	✓	✓	✓	
loglevel	✓ notice	✓ notice	✓ notice	✓ notice	
mailhostattrs	✓ mailHost	✓ mailHost	✓ mailHost		✓ mailHost
maxconcurrentconnectionattempts	✓ 32	✓ 32	✓ 32		
maxthreads	✓ 250				
numprocesses	✓ 1				
numthreads: deleted; see maxthreads					
plaintextmincipher		✓ 0	✓ 0		
polldelay	✓ -1				
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		✓ *long string—see text*			
servicelist: deleted; see tcp_listen options					
spoofemptymailbox			✓ 0		
spooftempfail			✓ 0		
spoofmessagefile			✓		
ssladjustciphersuites	✓	✓	✓	✓	✓
sslbacksideport		✓ 0	✓ 0		
sslcachedir	✓	✓	✓	✓	
sslcertnicknames: alias for sslnicknames					
sslcertprefix: deprecated; use base.ssldbprefix instead					
sslenable	✓ 0	✓ 0	✓ 0	✓ 0	
sslkeypasswdfile: deleted					
sslkeyprefix: deprecated; use base.ssldbprefix instead					
sslnicknames	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert	✓ Server-Cert
sslports: deleted					
sslsecmodfile: deleted					
storeadmin	✓ *complex*	✓ *complex*	✓ *complex*		✓ *complex*
storeadminpass	✓	✓	✓		✓
syncldap		✓ 1	✓ 1		
tcp_listen:*group-name.option-name*		✓	✓	✓	
tcpaccess	✓ *complex*	✓ *complex*	✓ *complex*	✓ *complex*	✓ *complex*
tcpaccessattr	✓ mailAllowedServiceAccess	✓ mailAllowedServiceAccess	✓ mailAllowedServiceAccess		✓ mailAllowedServiceAccess
timeout	✓ 1800	✓ 1800	✓ 1800	✓ 1800	
usergroupdn: deprecated; see base.ugldapbasedn instead					
use_nslog: deprecated	✓ 1	✓ 1	✓ 1	✓ 1	
usenslog: alias for use_nslog					
virtualdomaindelim	✓ *complex*	✓ *complex*	✓ *complex*		✓ *complex*
virtualdomainfile: deleted; see vdomain options instead					

See also the logfile options, which are available at many levels, including for the MMP and its components.

Note that many Base options are relevant to the MMP, including stressperiod and stressfdwait.

41.1 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.

41.2 adminpolicy Option

The adminpolicy MMP option specifies the algorithm used to determine which users have administrative proxy privilege when authenticating to the MMP. The default simple setting means there is only one administrator specified by the storeadmin option. The group setting causes the MMP to ignore the storeadmin option when determining who is and administrator and instead use the group-based admin policy used by the MTA and message store. In particular, proxy admins can be specified by the store.admins option, the store.indexeradmins option, the service administrators group as determined by the serviceadmingroupdn option, or a domain administrator group (a group within the domain with the inetMailAdministrator objectclass).

41.3 memcached_enable Option

The memcached_enable MMP option specifies whether badguy information is stored in memcached server or in process memory. The default is 0. If set to 1, stored in memcached.

41.4 memcached_host Option

The memcached_host MMP option specifies the memcached server host name. The default is local host.

41.5 memcached_port Option

The memcached_port MMP option specifies the port Number memcached listens to. The default port is 11211.

41.6 authcachettl Option Under the MMP

The Messaging Server can cache the results of successful LDAP authentication (*e.g.*, when logging into IMAP, or POP, or when the MMP has preauth enabled). The authcachettl

option (available at base, MMP, IMAP Proxy, POP Proxy, and vdomain levels) 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.

41.7 `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`.

41.8 `authenticationserver` Option

The `authenticationserver` Auth option specifies the hostname and port for a third-party authentication service to use for authentication. This option is also available at `imapproxy` and `popproxy` level (to override, for the respective server, the general Auth option). 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.

41.9 `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 is available at the `popproxy` and `vdomain` levels. 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.

41.10 `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; it may be set at the `popproxy` or `vdomain` level. 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.

41.11 `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. If

the value of backsideport option is same as sslbacksideport then the MMP will use SSL to connect to the message store server.

41.12 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))
```

41.13 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.

41.14 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.

41.15 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.

41.16 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.

41.17 bgmax Option

The bgmax option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the maximum number of IP addresses associated with authentication failures to keep track of simultaneously. See bgpenalty for more information.

41.18 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.

The bgpenalty option (available under base, imap, pop, mmp, imapproxy, and popproxy) specifies the length of time in seconds added to the authentication delay after each failed authentication.

41.19 bgmaxbadness Option

The bgmaxbadness option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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.

41.20 bgdecay Option

The bgdecay option (available under base, imap, pop, mmp, imapproxy, and popproxy) represents the time in seconds it takes for a BadGuy's penalty to be forgiven. See bgpenalty for more information.

41.21 bglinear Option

The bglinear option (available under base, imap, pop, mmp, imapproxy, and popproxy) defines whether a BadGuy's penalty decays linearly over time (1), or is a step function on expiration (0). See bgpenalty for more information.

41.22 bgexcluded Option

The bgexcluded option (available under base, imap, pop, mmp, imapproxy, and popproxy) 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).

41.23 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.

41.24 bindpass Option

Password the MMP uses in conjunction with the binddn option. DEPRECATED: Consider using ugldapbinddn and ugldapbindcred instead.

41.25 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.

41.26 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.

41.27 certmapdn Option

The legacy configuration certmapdn has been replaced in Unified Configuration by use of the more general usergroupdn option.

41.28 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.

41.29 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).

41.30 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).

41.31 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).

41.32 connlimits Option

The connlimits option (available under http, imap, pop, mmp, imapproxy, popproxy, 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 41.2 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".

The option has no default value; however, initial configuration normally sets a value of `:20` for the IMAP Proxy, and POP Proxy:

```
msconfig> show connlimits
role.imapproxy.connlimits = :20
role.popproxy.connlimits = :20
```

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.

41.32.1 connlimits Option Under isc_client

The `connlimits` isc_client option specifies the maximum number of connections that are permitted from a single LMTP process to the ISC server. Starting with 8.0.2.2, this isc_client option is deprecated; the `isc_client.max_conns` option should be used instead.

41.33 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.

41.34 crams Option

The `crams` option (available under `mmp`, `imapproxy`, `popproxy`, and `vdomain`) is a boolean indicating whether or not to enable the legacy 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.

Use of this option in new deployments is strongly discouraged as these authentication mechanisms provide poor security characteristics for the modern Internet.

41.35 debugkeys Option

The `debugkeys` option (available under `base`, `mmp`, `imapproxy`, `popproxy`, 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 41.3 Keywords for debugkeys option value

Keyword	Function
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)
dkim	diagnostics for built-in DKIM signing (new in 8.1.0.1)
dkimsig	perform an extra sanity check for built-in DKIM signing (new in 8.1.0.1) that is slower but may produce better diagnostics.
dkhash	log information about exactly how the DKIM hash is computed, including message content (may have privacy concerns).
dkimkey	log parsed DKIM private keys used for signing (may have privacy concerns).
dnsrbl	diagnostics for MMP DNS RBL function in main MMP log file (new in 8.0.2.3)
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).
eventloop	Log GDisp event loop statistics. May be helpful to diagnose performance problems (new in 8.0.2.2).
http	Log http transcripts related to Elasticsearch and ISC. (new in 8.0.2.2).
hula	log state changes in HULA (user lookup / authentication, new in 7.0.5)
ldap	log an LDAP directory protocol trace (replaces the now deprecated ldaptrace base option)
lpool	log ldap connection activity (mostly INFO & DEBUG level, new in 7.0.5); for MTA output, see also the os_debug MTA option
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
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)

`unicodembox`	enable debugging for unicode normalization of mailbox names (new in MS 8.0.2)

41.35.1 Use with base

The `debugkeys` base option specifies a space-separated list of keywords used to enable various optional debugging facilities; see Keywords That May Be Included in debugkeys Option Value.

Note that the SMTP server's `AUTH_DEBUG` TCP/IP-channel-specific option can override `debugkeys` for SMTP server authentication purposes.

For Message Store and other non-MTA processes, setting a relevant debugkey will enable NOTICE-level logging in the logfile for that process. The MTA has a different logging model and requires two additional settings to see debugging associated with a debugkey. First, MTA debug log files must be enabled (via `master_debug`, `slave_debug`, or the equivalent finer-grained mechanism). Second, it's necessary to set `mta.mm_debug` to a value of at least 3 for the DKIM-related debugkeys or to set `mta.os_debug` to 1 for the LDAP and authentication-related debugkeys.

41.36 `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.

41.36.1 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).

41.37 `dnsrbl` Option

The `dnsrbl` MMP option provides a list of domain names used to perform a DNS Realtime Blackhole List check for incoming POP and IMAP connections. The checks are performed in serial and may significantly impact the time it takes for clients to connect to the proxy. When a connection is rejected, a protocol-appropriate access denied error will be provided unless the connection is over SSL and the `sslconnlimit` option is also set, in which case an SSL `user_canceled` alert will be sent. If the `messagetrace.activate` option is set to `transactlog`, then the rejection will be noted in a `co` log entry in the store transaction log.

41.38 domainallowed Option Under the IMAP proxy

The domainallowed IMAP Proxy/POP Proxy option (also available under ens, eval_ldapd, http, imap, and pop) specifies access filters specifying which domains and/or IP addresses are allowed access for the selected server.

41.39 domainnotallowed Option Under the IMAP proxy

The domainnotallowed IMAP Proxy/POP Proxy option (also available at other levels) specifies access filters specifying which domains and/or IP addresses are not allowed access for the selected server.

41.40 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)
```

41.41 hosteddomains Option

The hosteddomains 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 hosteddomains 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), hosteddomains 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 hosteddomains 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.

41.42 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.

41.43 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.

41.44 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.

41.45 ldapcachesize Option

The MMP can cache results of user searches. The ldapcachesize option (available under mmp, imapproxy, popproxy, 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.

41.46 ldapcachettl Option

The MMP can cache results of user searches. The ldapcachesize option defines the number of cache entries; ldapcachettl (available under mmp, imapproxy, popproxy, and vdomain) 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.

41.47 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.

41.48 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.

41.49 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.

41.50 ldapurl Option

DEPRECATED as of 7.0.5: consider using ugldaphost, ugldapusessl, dcroot and ugldapbasedn instead.

The ldapurl option, available for the MMP, IMAP Proxy, and POP 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"
```

41.51 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 DATAROOT/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 DATAROOT/log. Changing that path to a soft-link is supported.

41.51.1 Use with isc

The logdir isc option specifies the log files location for ISC.

The user that owns the ISC process must have write permissions to this location.

41.51.2 Use with fit

The logdir fit option specifies the log files location for FIT.

The user that owns the DSE (Cassandra) process must have write permissions to this location.

41.52 loglevel Option Under the MMP

The MMP's loglevel option can be: nolog, emergency, alert, critical, error, warning, notice, information or debug. Note that loglevel will be ignored by the MMP if use_nslog is set to 0.

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'.

41.53 loglevel Option Under the IMAP proxy

The IMAP Proxy loglevel option can be: nolog, emergency, alert, crticial, error, warning, notice, information or debug. The MMP will not generate messages with priority higher than 'error'.

41.54 loglevel Option Under the POP proxy

The POP Proxy loglevel option can be: nolog, emergency, alert, crticial, error, warning, notice, information or debug. The MMP will not generate messages with priority higher than 'error'.

41.55 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.

41.56 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.

41.57 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.

41.58 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.

41.59 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.

41.60 plaintextmincipher Option Under the IMAP proxy

If the plaintextmincipher option under imapproxy or popproxy is > 0, then disable use of plaintext passwords over the respective 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.

41.61 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 noticably 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.

41.62 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.

41.63 preauthtimeout Option

The preauthtimeout option (available for the IMAP Proxy, POP Proxy) specifies the MMP session timeout prior to authentication.

41.64 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.

41.65 replayformat Option Under the MMP

The replayformat MMP/IMAP Proxy/POP Proxy/vdomain 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, prior to LDAP lookup)
%V (virtual domain only)
%A[attr] (value of user's attribute "attr")
```

For example, %A[uid]@%V for a user with joe as the value of the UID LDAP attribute 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.

41.66 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.

41.67 requireauthenticationserver Option

The requireauthenticationserver option is available under auth and under imapproxy and popproxy.

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.

41.68 restrictplainpasswords Option

When the restrictplainpasswords 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.

41.69 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)
```

41.70 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.

41.71 servicelist Option

For legacy MMP config only: The servicelist MMP option pecifies 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, and imapproxy sections.

41.72 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.

41.73 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.

41.74 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.

41.75 ssladjustciphersuites Option

The ssladjustciphersuites option allows adjusting which SSL cipher suites are enabled or disabled. (This option is available under base, mmp, imapproxy, popproxy, and vdomain.) 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 crytographic 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 will 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 cipher suites that require installation of special server certificate types, for example the DSS cipher suites. Such a site would enable these additional suites once installation was complete.

3. If a site is forced to continue supporting a particularly old client that only supports old cipher suites, they can be explicitly enabled (for example '+RC4' enables the RC4 cipher suites).

4. A site that chooses to disable an older cipher or hash function pro-actively despite potential interoperability issues may choose to do so. For example, to disable all ciphers using the '3DES' or 'SHA1' algorithms, simply set '-3DES,-SHA1'. Be aware that this sort of pro-active action may generate support calls from end users running older mail clients.

5. 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 '3DES' algorithm, simply set '-3DES'.

As of NSS 3.28 (2017), the following cipher suites are enabled by default in the NSS library: TLS_AES_128_GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256, TLS_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_3DES_EDE_CBC_SHA.

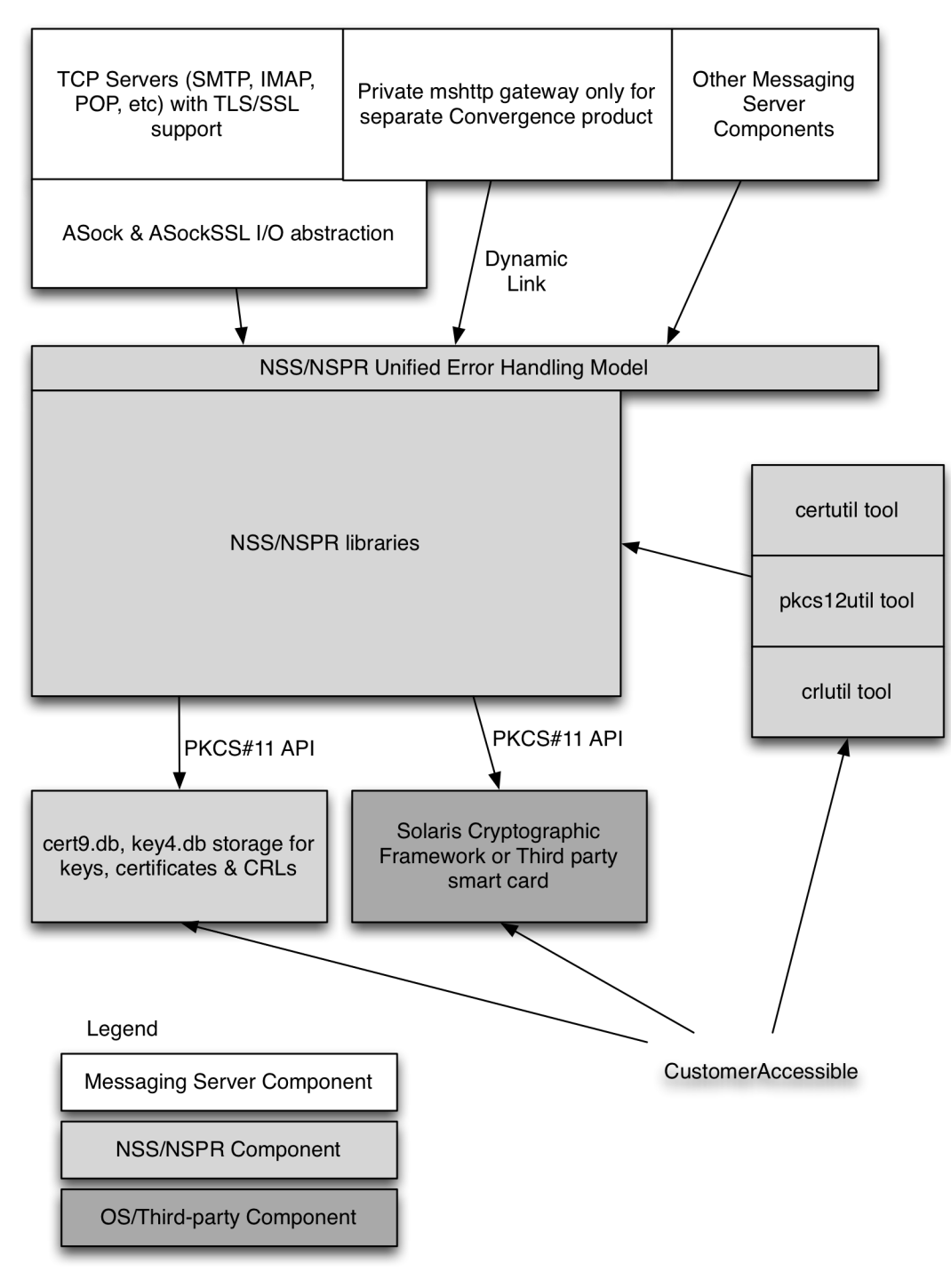
The complete list of cipher suites present in NSS 3.28 (2017) follows:
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_DES_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA, SSL_DHE_RSA_WITH_DES_CBC_SHA,
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5, SSL_RSA_EXPORT_WITH_RC4_40_MD5,
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA, SSL_RSA_FIPS_WITH_DES_CBC_SHA,
SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_RSA_WITH_DES_CBC_SHA,
SSL_RSA_WITH_NULL_MD5, SSL_RSA_WITH_NULL_SHA,
SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA,
TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384,
TLS_CHACHA20_POLY1305_SHA256, TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_RC4_128_SHA, TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_NULL_SHA, TLS_ECDHE_ECDSA_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_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,

TLS_ECDHE_RSA_WITH_NULL_SHA, TLS_ECDHE_RSA_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_ECDH_ECDSA_WITH_NULL_SHA,
TLS_ECDH_ECDSA_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_ECDH_RSA_WITH_NULL_SHA, TLS_ECDH_RSA_WITH_RC4_128_SHA,
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA,
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256,
TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384,
TLS_RSA_WITH_NULL_SHA256, TLS_AES_128_GCM_SHA256,
TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256,
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256, TLS_DHE_DSS_WITH_AES_256_CBC_SHA,
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384,
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA,
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA,
TLS_DHE_DSS_WITH_RC4_128_SHA, TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA,
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA,
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_ECDSA_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_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,
TLS_ECDHE_RSA_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_ECDH_ECDSA_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_ECDH_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA,

TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_RSA_WITH_CAMELLIA_128_CBC_SHA, TLS_RSA_WITH_CAMELLIA_256_CBC_SHA, TLS_RSA_WITH_RC4_128_MD5, TLS_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_SEED_CBC_SHA.

TLS 1.3 (NSS 3.39+) has its own cipher suites separate from previous TLS versions. These include: TLS_AES_128_GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256, TLS_AES_256_GCM_SHA384.

Always keep in mind that adjusting the available ciphersuites can impact multiple protocols; NSS is tightly integrated into Messaging Server, as the following diagram illustrates:



41.76 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.)

41.77 sslcachedir Option

The sslcachedir option, (available under base, mmp, imapproxy, and popproxy), 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 predictable, this option must be the same for the IMAP Proxy and POP Proxy -- any settings of this option for the proxies must match. (Or better yet, don't set it explicitly for any of the proxies; instead set it at MMP level.)

41.78 sslcertprefix Option

The sslcertprefix option (available under mmp, imapproxy, and popproxy) specifies the 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 predictable, this option must be the same for the IMAP and POP proxies.

41.79 sslenable Option

The sslenable option, available for the MMP, IMAP Proxy, and POP Proxy, specifies whether SSL is enabled for the specified proxy service (via the STARTTLS command).

41.80 sslkeypasswdfile Option

DELETED as of MS 7.0.5.

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 ssldbpath to relocate these files instead.

NOTE: In order for results to 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.

41.81 sslkeyprefix Option

Filename prefix to the SSL key database file. The key 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 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.

41.82 sslnicknames Option Under the MMP

The sslnicknames option (available at MMP, IMAP Proxy, POP Proxy, and vdomain level, as well as at Base and other 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.

41.83 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.

41.84 storeadmin Option

The 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.

41.85 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. If not set, this defaults to the value of proxyadminpass.

41.86 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.

41.87 tcp_listen options

Named tcp_listen groups can appear under the imapproxy, and popproxy groups. Several options may be set under tcp_listen.

41.87.1 tcp_ports Option Under tcp_listen

Under a top level imapproxy or popproxy 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
```

41.87.2 ssl_ports Option Under tcp_listen

Under a top level imapproxy or popproxy 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
```

41.87.3 listen_addresses Option Under tcp_listen

Under a top level imapproxy or popproxy 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).

41.87.4 backlog Option Under tcp_listen

RESTRICTED: Under a top level imapproxy or popproxy 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.

41.88 tcpaccess Option

The tcpaccess option (available under mmp, imapproxy, popproxy, 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 TCP wrapper filter syntax.

41.89 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.

41.90 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.

See also the options for the IMAP server, POP server, and SMTP SUBMIT server: imap.idletimeout, pop.idletimeout, and the SMTP SUBMIT server's various *_TIME channel-specific options.

41.91 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.

See also the IMAP server's own idletimeout option.

41.92 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.

See also the POP server's own idletimeout option.

41.93 use_nslog Option Under the MMP

DEPRECATED for the MMP and its components as of MS 8.0: The use_nslog option (available at levels including mmp, imapproxy, and popproxy) 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.

41.94 usenslog Option

The legacy configuration usenslog has been replaced in Unified Configuration by use_nslog.

41.95 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 ugldapbasedn option.

41.96 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.

41.97 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.

Part VI 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, MSHTTP may call out to ICAP to perform HTML sanitization; this support has a number of icapservice configuration options.

For Messenger Express, see also the `rfc822headerallow8bit` base option.

Chapter 42 MSHTTP options

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.

42.1 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.

42.2 enablesslport Option Under http

Sets whether or not the HTTP over SSL service for Convergence is started. If enabled, the HTTP+SSL service listens on the port specified by the sslport MSHTTP option. For the 7.0.5 release, the sslusessl MSHTTP 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 MSHTTP option.

Note that 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.

42.3 allowanonymouslogin Option Under http

The allowanonymouslogin MSHTTP option enables the SASL ANONYMOUS mechanism for mshttpd.

42.4 allowcollect Option

Set the allowcollect MSHTTP option to 0 to prevent the server from performing remote POP mailbox collection.

Note that the maxcollectmsglen MSHTTP option specifies a maximum on the message size that may be collect, while the popbindaddr MSHTTP option specifies the IP address to which to bind outgoing POP connections when collecting external mail.

42.5 allowldapaddresssearch Option

The allowldapaddresssearch MSHTTP option controls whether legacy webmail client users (*i.e.*, users of Messenger Express & Communications Express) can search the directory for addresses.

42.6 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.

42.7 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.

42.8 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; the default is 55443.

42.9 charsetvalidation Option

Set the charsetvalidation MSHTTP 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.

42.10 connlimits Option

The connlimits option (available under http, imap, pop, mmp, imapproxy, popproxy, 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 42.1 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".

The option has no default value; however, initial configuration normally sets a value of :20 for the IMAP Proxy, and POP Proxy:

```
msconfig> show connlimits
role.imapproxy.connlimits = :20
role.popproxy.connlimits = :20
```

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.

42.10.1 connlimits Option Under isc_client

The connlimits isc_client option specifies the maximum number of connections that are permitted from a single LMTP process to the ISC server. Starting with 8.0.2.2, this isc_client option is deprecated; the isc_client.max_conns option should be used instead.

42.11 convergencefilterenabled Option

Starting with MS 8.0.2, mshttpd disables the built-in security filter for HTML and only has a filter to process URLs. As a result, use of Convergence version 3.0.1.1.0 or later with the modern whitelist HTML filter it provides is recommended. Once you have verified your Convergence version is compatible, and you have verified that the Convergence option mail.htmlsanitizer.enable is on, then you may set this option to 1. If you set this option to 1 without first verifying correct Convergence version and configuration, your Convergence end-users will be vulnerable to a number of security attacks.

42.12 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.

42.13 cookiename Option

The cookiename MSHTTP 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 MSHTTP 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.

42.14 da_host Option

The da_host MSHTTP option specifies the 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.

42.15 da_port Option

The da_port MSHTTP option specifies the Delegated Administrator 1.x port, by default 8080. This option is available for backwards compatibility only, and should no longer be used.

42.16 detectcharset Option

The detectcharset option for mshttpd enables automatic character set detection for unlabeled text parts supplied by Convergence.

42.17 domainallowed Option Under http

The domainallowed MSHTTP option specifies access filters specifying which domains and/or IP addresses are allowed HTTP access.

42.18 domainnotallowed Option Under http

The domainnotallowed MSHTTP option specifies access filters specifying which domains and/or IP addresses are not allowed HTTP access.

42.19 enableblacklistfilter Option

Starting with MS 8.0.2, mshttpd disables the built-in blacklist HTML filter by default and only has a filter to process URLs. Use of the modern whitelist HTML sanitizer available in Convergence 3.0.1.1.0 or later is recommended. This option can be set to 1 if the Convergence mail.htmlsanitizer.enable option is set. This can also be used if a deployment is unable to upgrade Convergence and is comfortable with the legacy blacklist filter.

42.20 enableuserlist Option Under http

The enableuserlist MSHTTP option enables imsconnutil connected user listing for HTTP service.

42.21 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).

42.22 filterhiddenmailinglists Option

The filterhiddenmailinglists MSHTTP option excludes the mgmanhidden LDAP attribute from the search filter when set to 0.

42.23 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.

42.24 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.

42.25 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.

42.26 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.

42.27 gzipattach Option

The gzipattach MSHTTP option enables (when set to 1) or disables attachment download gzip by default for Internet Explorer clients.

42.28 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.

42.29 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.

42.30 htmlprocessor Option

The htmlprocessor MSHTTP option controls how html mail is sanitized prior to display by the Convergence webmail client. The mshttpd built-in blacklist processor is used when the value is set to 0 or 1. In MS 8.0.1, setting this option to 2 enables use of the ICAP service for HTML sanitization.

Use of the white-list sanitizer available in Convergence 3.0.1.1.0 or later is recommended. That filter is enabled by setting the Convergence option mail.htmlsanitizer.enable.

Starting with MS 8.0.2, mshttpd disables the built-in blacklist filter by default and only has a filter to process URLs. As a result, Convergence 3.0.1.1.0 or later is strongly recommended

with MS 8.0.2 and the Convergence option `mail.htmlsanitizer.enable` should be on. See the `http.convergencefilterenabled` option.

42.31 `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.

42.32 `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.

42.33 `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.

42.34 `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.

42.35 `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.

42.36 `ldapaddresssearchattrs` Option

The `ldapaddresssearchattrs` MSHTTP option specifies a string containing a comma-delineated list of LDAP attributes returned to legacy webmail client users (*i.e.*, users of Messenger Express or Communications Express) in a directory search.

42.37 `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.

42.38 `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.

Note that the allowcollect MSHTTP option controls whether or not such external collection is permitted, while the popbindaddr MSHTTP option specifies the IP address to which to bind outgoing POP connections when collecting external mail.

42.39 maxldaplimit Option

The maxldaplimit MSHTTP option sets the maximum LDAP lookup limit.

42.40 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.

See also the maxpostsize MSHTTP option.

42.41 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, http.maxmessagesize). In legacy configuration this would be max(5*1024*1024, service.http.maxmessagesize).

42.42 maxsessions Option Under http

The maxsessions MSHTTP option specifies the maximum number of sessions per MSHTTP server process.

42.43 maxthreads Option Under http

The maxthreads MSHTTP option specifies the maximum number of threads per MSHTTP server process.

42.44 nofilecache Option

The nofilecache MSHTTP option disables html files caching; used for debugging.

42.45 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.

42.46 plaintextconvspace Option

If the plaintextconvspace MSHTTP option is set to "1", spaces in text messages will be converted to non-breaking spaces in webmail.

42.47 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.

42.48 plaintexttabsize Option

The plaintexttabsize MSHTTP option sets the tabsize for text message display in webmail.

42.49 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.

Note that the allowcollect MSHTTP option controls whether or not such external collection is permitted, and the maxcollectmsglen MSHTTP option specifies a maximum on the message size that may be collected.

42.50 port Option Under http

The port MSHTTP option specifies the Messenger Express HTTP port. The default is 8990.

42.51 proxyport Option

The proxyport MSHTTP option configures the port number of the back-end Messenger Express (HTTP) server with the Messaging Multiplexor.

42.52 replayformat Option Under http

The replayformat MSHTTPD option, http.replayformat, specifies the format for authentication replay from mshttpd to IMAP and MTA backends. Supports:

- %o for original userid as sent by the client,
- %s for *user@domain*,
- %U for userid only (prior to LDAP lookup),
- %V for virtual domain,
- %A[*attr*] for value of specified user's attribute.

42.53 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, sessiontimeout/2), thus normally 900, (corresponding to 15 minutes), unless sessiontimeout has been set to an unusually low value.

42.54 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.

42.55 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.

42.56 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.

42.57 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.

42.58 smtpauthpassword Option Under http

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.

42.59 smtpauthuser Option Under http

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.

42.60 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.

42.61 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.

42.62 smtptls Option Under http

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.

42.63 sourceurl Option

The sourceurl MSHTTP option specifies the URL of the webmail server.

42.64 spooldir Option

The spooldir MSHTTP option specifies the attachment spool directory for client outgoing mail; if unspecified, uses the directory specified in the tmpdir Base option (in legacy configuration, local.tmpdir).

42.65 sslcachesize Option Under http

The sslcachesize MSHTTP option specifies the number of SSL sessions to be cached by the MSHTTP server. If this is set to 0 or not set, this will use a default provided by the Mozilla NSS library which was 10000 last time this was checked (March 2016).

42.66 sslnicknames Option Under 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).

42.67 sslport Option Under http

The sslport MSHTTP option, http.sslport, specifies the port number for the HTTP over SSL service. The default is 8891. Note that to enable the HTTP+SSL service, the enablesslport MSHTTP option must also be set. (In MS 7.0.5, the sslusessl MSHTTP option has to be explicitly set to 1 as well, even though the default was 1; as of MS 8.0, the sslusessl option is not needed, and indeed is ignored, by MSHTTP.)

42.68 sslsourceurl Option

The sslsourceurl MSHTTP option specifies the URL of the webmail server when SSL is enabled. (For enabling SSL, see the enablesslport MSHTTP option; note that in order for the sslsourceurl URL to be used, not only must enablesslport be enabled, but the SSL initialization must indeed have succeeded.)

42.69 sslusessl Option Under http

Starting with the 8.0 release, this option has no effect on the mshttpd server.

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 http.enablesslport to enable SSL connections on a separate sslport.

42.70 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.

42.71 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.

42.72 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.

42.73 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.

42.74 uwcenabled Option

The `uwcenabled` 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 `uwcenabled` and `cookiename` (`local.webmail.sso.uwcenabled` and `local.service.http.cookiename` in legacy configuration) must match on the front and back ends.

42.75 uwccontexturi Option

The `uwccontexturi` 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.uwccontexturi=uwc`.

42.76 uwchome Option

The `uwchome` MSHTTP option specifies the URL required to access the home link.

42.77 uwclogouturl Option

The `uwclogouturl` MSHTTP option specifies the URL Messenger Express uses to invalidate the Communications Express session.

42.78 uwcport Option

The uwcport MSHTTP option specifies the Communications Express port.

42.79 uwcsslport Option

The uwcsslport MSHTTP option specifies the Communications Express SSL port.

42.80 xmailer Option

The xmailer MSHTTP option may be set to override the X-Mailer: header field value with the specified string.

42.81 MSHTTP errors

To localize/customize MSHTTP error messages:

1. Copy the files *SERVERROOT*/lib/bundles/http_err_*.properties to a site-private (not a Messaging Server) directory.
2. Modify the files as desired.
3. Use the genrb utility to generate a new .res file.
4. Copy the site-generated .res file back to the *SERVERROOT*/lib/bundles/http_err_*.properties

Note that the lib/bundles directory can be rewritten by patches; check for changes to the files with any patches and version updates, so that you can reinstantiate the site-specific changes.

42.82 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.

42.82.1 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.

42.82.2 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.

42.83 MSHTTP httpcharset and mailcharset options

Under http are httpcharset and mailcharset options to set the (default) character set for various languages.

42.83.1 af Option

The af option (available under http.httpcharset and http.mailcharset) sets the character set used for Afrikaans.

42.83.1.1 Use with httpcharset

Default http character set for Afrikaans.

42.83.1.2 Use with mailcharset

Default mail character set for Afrikaans.

42.83.2 ar Option

42.83.2.1 Use with httpcharset

Default http character set for Arabic.

42.83.2.2 Use with mailcharset

Default mail character set for Arabic.

42.83.3 be Option

42.83.3.1 Use with httpcharset

Default http character set for Byelorussian.

42.83.3.2 Use with mailcharset

Default mail character set for Byelorussian.

42.83.4 bg Option

42.83.4.1 Use with httpcharset

Default http character set for Bulgarian.

42.83.4.2 Use with mailcharset

Default mail character set for Bulgarian.

42.83.5 ca Option

42.83.5.1 Use with httpcharset

Default http character set for Catalan.

42.83.5.2 Use with mailcharset

Default mail character set for Catalan.

42.83.6 cs Option

42.83.6.1 Use with httpcharset

Default http character set for Czech.

42.83.6.2 Use with mailcharset

Default mail character set for Czech.

42.83.7 da Option

42.83.7.1 Use with httpcharset

Default http character set for Danish.

42.83.7.2 Use with mailcharset

Default mail character set for Danish.

42.83.8 de Option

42.83.8.1 Use with httpcharset

Default http character set for German.

42.83.8.2 Use with mailcharset

Default mail character set for German.

42.83.9 el Option

42.83.9.1 Use with httpcharset

Default http character set for Greek.

42.83.9.2 Use with mailcharset

Default mail character set for Greek.

42.83.10 en Option

42.83.10.1 Use with httpcharset

Default http character set for English.

42.83.10.2 Use with mailcharset

Default mail character set for English.

42.83.11 es Option

42.83.11.1 Use with httpcharset

Default http character set for Spanish.

42.83.11.2 Use with mailcharset

Default mail character set for Spanish.

42.83.12 et Option

42.83.12.1 Use with httpcharset

Default http character set for Estonian.

42.83.12.2 Use with mailcharset

Default mail character set for Estonian.

42.83.13 eu Option

42.83.13.1 Use with httpcharset

Default http character set for Basque.

42.83.13.2 Use with mailcharset

Default mail character set for Basque.

42.83.14 fi Option

42.83.14.1 Use with httpcharset

Default http character set for Finnish.

42.83.14.2 Use with mailcharset

Default mail character set for Finnish.

42.83.15 fr Option

42.83.15.1 Use with httpcharset

Default http character set for French.

42.83.15.2 Use with mailcharset

Default mail character set for French.

42.83.16 ga Option

42.83.16.1 Use with httpcharset

Default http character set for Irish.

42.83.16.2 Use with mailcharset

Default mail character set for Irish.

42.83.17 gl Option

42.83.17.1 Use with httpcharset

Default http character set for Galician.

42.83.17.2 Use with mailcharset

Default mail character set for Galician.

42.83.18 he Option

42.83.18.1 Use with httpcharset

Default http character set for Hebrew.

42.83.18.2 Use with mailcharset

Default mail character set for Hebrew.

42.83.19 hr Option

42.83.19.1 Use with httpcharset

Default http character set for Croatian.

42.83.19.2 Use with mailcharset

Default mail character set for Croatian.

42.83.20 hu Option

42.83.20.1 Use with httpcharset

Default http character set for Hungarian.

42.83.20.2 Use with mailcharset

Default mail character set for Hungarian.

42.83.21 is Option

42.83.21.1 Use with httpcharset

Default http character set for Icelandic.

42.83.21.2 Use with mailcharset

Default mail character set for Icelandic.

42.83.22 it Option

42.83.22.1 Use with httpcharset

Default http character set for Italian.

42.83.22.2 Use with mailcharset

Default mail character set for Italian.

42.83.23 ja Option

42.83.23.1 Use with httpcharset

Default http character set for Japanese.

42.83.23.2 Use with mailcharset

Default mail character set for Japanese.

42.83.24 ko Option

42.83.24.1 Use with httpcharset

Default http character set for Korean.

42.83.24.2 Use with mailcharset

Default mail character set for Korean.

42.83.25 lt Option

42.83.25.1 Use with httpcharset

Default http character set for Lithuanian.

42.83.25.2 Use with mailcharset

Default mail character set for Lithuanian.

42.83.26 lv Option

42.83.26.1 Use with httpcharset

Default http character set for Latvian.

42.83.26.2 Use with mailcharset

Default mail character set for Latvian.

42.83.27 mk Option

42.83.27.1 Use with httpcharset

Default http character set for Macedonian.

42.83.27.2 Use with mailcharset

Default mail character set for Macedonian.

42.83.28 nl Option

42.83.28.1 Use with httpcharset

Default http character set for Dutch.

42.83.28.2 Use with mailcharset

Default mail character set for Dutch.

42.83.29 no Option

42.83.29.1 Use with httpcharset

Default http character set for Norwegian.

42.83.29.2 Use with mailcharset

Default mail character set for Norwegian.

42.83.30 pl Option

42.83.30.1 Use with httpcharset

Default http character set for Polish.

42.83.30.2 Use with mailcharset

Default mail character set for Polish.

42.83.31 pt Option

42.83.31.1 Use with httpcharset

Default http character set for Portuguese.

42.83.31.2 Use with mailcharset

Default mail character set for Portuguese.

42.83.32 ro Option

42.83.32.1 Use with httpcharset

Default http character set for Romanian.

42.83.32.2 Use with mailcharset

Default mail character set for Romanian.

42.83.33 ru Option

42.83.33.1 Use with httpcharset

Default http character set for Russian.

42.83.33.2 Use with mailcharset

Default mail character set for Russian.

42.83.34 sk Option

42.83.34.1 Use with httpcharset

Default http character set for Slovak.

42.83.34.2 Use with mailcharset

Default mail character set for Slovak.

42.83.35 sl Option

42.83.35.1 Use with httpcharset

Default http character set for Slovenian.

42.83.35.2 Use with mailcharset

Default mail character set for Slovenian.

42.83.36 sq Option

42.83.36.1 Use with httpcharset

Default http character set for Albanian.

42.83.36.2 Use with mailcharset

Default mail character set for Albanian.

42.83.37 sr Option

42.83.37.1 Use with httpcharset

Default http character set for Serbian.

42.83.37.2 Use with mailcharset

Default mail character set for Serbian.

42.83.38 sv Option

42.83.38.1 Use with httpcharset

Default http character set for Swedish.

42.83.38.2 Use with mailcharset

Default mail character set for Swedish.

42.83.39 th Option

42.83.39.1 Use with httpcharset

Default http character set for Thai.

42.83.39.2 Use with mailcharset

Default mail character set for Thai.

42.83.40 tr Option

42.83.40.1 Use with httpcharset

Default http character set for Turkish.

42.83.40.2 Use with mailcharset

Default mail character set for Turkish.

42.83.41 uk Option

42.83.41.1 Use with httpcharset

Default http character set for Ukrainian.

42.83.41.2 Use with mailcharset

Default mail character set for Ukrainian.

42.83.42 yi Option

42.83.42.1 Use with httpcharset

Default http character set for Yiddish.

42.83.42.2 Use with mailcharset

Default mail character set for Yiddish.

42.83.43 zh-cn Option

42.83.43.1 Use with httpcharset

Default http character set for Simplified Chinese.

42.83.43.2 Use with mailcharset

Default mail character set for Simplified Chinese.

42.83.44 zh-tw Option

42.83.44.1 Use with httpcharset

Default http character set for Traditional Chinese.

42.83.44.2 Use with mailcharset

Default mail character set for Traditional Chinese.

42.84 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`.

42.84.1 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.

42.84.2 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 43 SMIME options

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`.

43.1 `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.

43.2 `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))
```

43.3 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)
```

43.4 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?
```

43.5 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)
```

43.6 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

43.7 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.

43.8 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;
```

43.9 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;
```

43.10 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;
```

43.11 platformhpux Option

The platformhpux S/MIME option specifies one or more library names needed by smart cards or local key store on the HPUX 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 HPUX platform. Example:

```
MOZILLA:library=libsoftokn3.sl;
```

43.12 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;
```

43.13 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 43.1 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.

43.14 alwayssign Option

The alwayssign 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 43.2 Possible alwayssign 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.

43.15 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 43.3 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.

43.16 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 DATAROOT/store/mboxlist.

43.17 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

43.18 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.

43.19 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)

43.20 timestampdelta Option

The timestampdelta S/MIME option specifies the time interval in seconds before the received time on messages.

43.21 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 43.4 crlaccessfail fields

Field	Meaning
#fails	number of CRL URL failures before triggering wait
#mins	number of minutes during which the #fails must occur
#minswait	number of minutes to wait until another CRL URL attempt will be allowed If crlaccessfail is used, no fields are optional, and all values must be greater than 0.

Example: 2:20:10.

43.22 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:

Table 43.5 checkoverssl S/MIME option values

Value	Meaning
0	do not use an SSL communications link.
1	use an SSL communications link. This is the default.

43.23 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 43.6 readsigncert fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

43.24 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:

Table 43.7 revocationunknown S/MIME option Values

Value	Meaning
ok	treat the certificate as valid.
revoked	treat the certificate as revoked. This is the default.

43.25 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 43.8 sendencryptcert fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

43.26 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 43.9 sendencryptedcertrevoked fields

Value	Meaning
allow	use the certificate.
disallow	do not use the certificate. This is the default.

43.27 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 43.10 sendsigncert fields

Value	Meaning
0	do not check the certificate against a CRL.
1	check the certificate against a CRL. This is the default.

43.28 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 43.11 sendsigncertrevoked fields

Value	Meaning
allow	use the certificate.
disallow	do not use the certificate. This is the default.

43.29 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 43.12 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.

43.30 appletlogging Option

The appletlogging S/MIME option specifies whether to enable logging on the applet. Choose one of these values:

Table 43.13 appletlogging fields

Value	Meaning
0	do not log applet output. This is the default.
1	log applet output.

Chapter 44 SSO options

The `verifyurl` option is the only option under the `sso` group. See also various MSHTTP options relating to SSO, especially the `sso_*` MSHTTP options.

44.1 `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 45 icapservice options

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.

45.1 `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.

45.2 `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.

45.3 `server_host` Option Under icapservice

The `server_host icapservice` option specifies the ICAP server host name. If empty or not set, the loopback interface is used.

45.4 `server_port` Option Under icapservice

The `server_port icapservice` option specifies the ICAP server port. The default is `1344`.

Part VII 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 requeuing 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 46 Channels

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).

The execution of channel programs is primarily triggered and controlled by the two major "control" processes of the MTA: the Dispatcher listens on TCP ports and triggers execution of appropriate "slave" channel programs (such as the SMTP server and LMTP server "slave" programs), while the Job Controller maintains a database of what messages are awaiting delivery attempts, and schedules and triggers execution of channel programs (primarily channel "master" programs) to attempt deliveries. However, channel programs can be run via other mechanisms; see, for instance, the `imsimta run` utility, used for manual triggering of channel program execution.

46.1 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 46.1 Modern reserved channel names

Name	Reserved For
`bitbucket`	Bit bucket channel (deletes all messages queued to it)
`bsin_*`	BSMTP inbound channels
`bsout_*`	BSMTP outbound channels
`circuitcheck`	Message circuit checking channel
`conversion`	Message body part conversion channel
`defragment`	Message defragmentation channel
`filter_discard`	Channel for delayed deletion of message discarded or jettisoned due to Sieve actions (or analogous `*_ACCESS` mapping table flag effects)
`hold`	Channel where messages are temporarily detained for administrative purposes such as user migration

ims-ms	Channel delivering to the Message Store (without use of LMTP; see also the tcp_lmtpss channel when LMTP is used instead).
ims-ms_*	Additional channels delivering to the Message Store.
l	The local channel; in modern Messaging Server configurations, a "placeholder" channel that performs no delivery but rather performs alias processing only.
native	Delivery to UNIX /var/mail mailboxes.
pipe	Pipe channel
pipe_*	Additional pipe channels
process	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 46.2 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
dn_	PhoneNet over DECnet
dsmtp_	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
xsmtp_	SMTP over X.25; obsolete

46.2 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.identnonenumeric (novalue)
role.channel:tcp_local.inner (novalue)
role.channel:tcp_local.loopcheck (novalue)
role.channel:tcp_local.maysaslserver (novalue)
role.channel:tcp_local.maytlsserver (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 identnonenumeric pool SMTP_POOL \
  switchchannel maysaslserver saslswitchchannel tcp_auth maytlsserver \
  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.

46.2.1 defaults and nodefaults pseudo-channels

Many configurations involve repetition of various channel options on all or nearly all channels. Maintaining such a configuration is both tedious and error-prone. The MTA offers a simple way to change what options are set by default on various channels. This mechanism can be used to greatly simplify some configurations.

Note that in Unified Configuration channel neither multiple defaults channel nor the relative ordering of channels in the configuration are preserved. However, care is taken when the configuration is read to process the defaults channel first. As such, unified configuration is limited to having a single defaults channel that affects all subsequent channels.

If a line of the form:

```
defaults option1 option2 option3 ...
```

is inserted into the configuration, all channel blocks following this line will inherit the options specified on the line. The `defaults` line can be thought of as a special channel block that changes the option defaults without actually specifying a channel. The `defaults` line, or pseudo-channel definition, also does not require any additional lines of channel block information (if any are specified they will be ignored); in Unified Configuration terms, the `defaults` pseudo-channel does not need (and will ignore) any `official_host_name`, `local_host_alias`, or `additional_host_names` option settings.

There is no limit on the number of `defaults` lines that can be specified in a legacy configuration --- the effects of multiple `defaults` lines are cumulative with the most recently encountered (reading from top to bottom) line having precedence. Unified configurations are limited to a single defaults channel.

It may be useful to unconditionally eliminate the effects of any `defaults` lines starting at some point in a legacy configuration file (at the start of a standalone section of channel blocks in an external file, for example). The `nodefaults` line is provided for this purpose. It takes the form:

```
nodefaults
```

and has the obvious effect --- it nullifies all settings established by any previous `defaults` channel and returns the configuration to the state that would apply if no `defaults` had been specified. This functionality is not available in unified configurations.

Like regular channel blocks, a blank line must separate each `defaults` or `nodefaults` channel block from other channel blocks. The `defaults` and `nodefaults` channel blocks are the only channel blocks which may appear before the local channel in the configuration file. However, like any other channel block, they must appear after the last rewrite rule.

Initial configuration generates a `defaults` pseudo-channel located prior to any other channel definition (so applying to all other initially configured channels):

```
msconfig> show channel:defaults.*
role.channel:defaults.defaulthost = &/IMTA_DEFAULTDOMAIN/ &/IMTA_DEFAULTDOMAIN/
role.channel:defaults.maxjobs = 7
role.channel:defaults.noswitchchannel (novalue)
role.channel:defaults.notices = 1 2 4 7
```

46.3 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` 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` 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.

46.3.1 Alphabetic list of channel options

Channel options listed alphabetically below lists channel keywords alphabetically. Channel options shown in **`bold face type`** are defaults.

Table 46.3 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 MS 6.1) Accept messages despite certain errors that would normally cause message rejection
`accepttemporaryfailures`	(New in 8.0.1.1.0) Accept messages despite recipient address temporary error conditions (*e.g.*, over quota, LDAP server unavailable, Spam/virus filter unavailable, *etc.*).
`acceptvalidaddresses`	(New in MS 6.1) Perform normal rejection checks on incoming messages
`additional_host_names`	Additional hosts the channel can reach
`addlineaddrs`	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
`addrsperfile`	Number of addresses per message file
`addrsperjob`	Number of addresses to be processed by a single job
`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 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 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`
`authhost`	(New in 8.0.2.1) Use the domain of the authenticated user's primary address complete addresses
`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`
`bccserver`	(New in MS 8.0.2.3) XBCC extension is enabled
`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
conditionalsecuritymultiparts	Process inside security multiparts, 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	DEPRECTED: Specify folding point in batch SMTP lines
convertoctetstream	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
defaulthost	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
defertemporaryfailures	(New in 8.0.1.1.0) Defer, with SMTP temporary rejection, recipient addresses with temporary error conditions (*e.g.*, over quota, LDAP server unavailable, Spam/virus filter unavailable, *etc.*).
defragment	Reassemble any MIME-compliant message/partial parts queued to this channel
deletemessagehash	(New in MS 6.3) Delete message hash
deliveryflags	Set flags controlling certain delivery behaviors
dequeue_removeroute	Alias for dequeueremoveroute; in legacy configuration, strip source route (@source-route:address) when dequeuing message
dequeueremoveroute	(Unified Configuration only) Strip source route (@source-route:address) when dequeuing message
description	Channel description
destinationbrightmail	DEPRECATED: Alias for destinationspamfilter1
destinationbrightmailoptin	DEPRECATED: Alias for destinationspamfilter1optin
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
destinationnosolicit	New in MS 6.2. List of solicitation types not accepted
destinationspamfilter	Alias for destinationspamfilter1
destinationspamfilter1	(New in MS 6.2) Enable spam/virus filtering (by spam/filter package 1) for messages enqueued to this destination channel
destinationspamfilter1optin	(New in 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
destinationspamfilter2	(New in MS 6.2) Enable spam/virus filtering (by spam/filter package 2) for messages enqueued to this destination channel
destinationspamfilter2optin	(New in 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
destinationspamfilter3	(New in MS 6.2) Enable spam/virus filtering (by spam/filter package 3) for messages enqueued to this destination channel
destinationspamfilter3optin	(New in 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

destinationspamfilter4	(New in MS 6.2) Enable spam/virus filtering (by spam/filter package 4) for messages enqueued to this destination channel
destinationspamfilter4optin	(New in 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
destinationspamfilter5	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 5) for messages enqueued to this destination channel
destinationspamfilter5optin	(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
destinationspamfilter6	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 6) for messages enqueued to this destination channel
destinationspamfilter6optin	(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
destinationspamfilter7	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 7) for messages enqueued to this destination channel
destinationspamfilter7optin	(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
destinationspamfilter8	(New in MS 6.3) Enable spam/virus filtering (by spam/filter package 8) for messages enqueued to this destination channel
destinationspamfilter8optin	(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
destinationspamfilteroptin	Synynom for destinationspamfilter1optin
destinationsrs	(New in MS 6.3p1) Messages destined out this channel are eligible for SRS encoding
disabledestinationbrightmail	DEPRECATED: Alias for disabledestinationspamfilter1
disabledestinationfilter	(New in MS 7.0u3) Disable evaluation and application of specified Sieve filters
disabledestinationspamfilter	Alias for disabledestinationspamfilter1
disabledestinationspamfilter1	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 1) for messages enqueued to this channel
disabledestinationspamfilter2	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 2) for messages enqueued to this channel
disabledestinationspamfilter3	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 3) for messages enqueued to this channel
disabledestinationspamfilter4	(New in 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	DEPRECATED: Alias for disablesourcespamfilter1
disablesourcefilter	(New in MS 7.0u3) Disable evaluation and application of specified Sieve filters
disablesourcespamfilter	(New in MS 6.1) Synynom for disablesourcespamfilter1
disablesourcespamfilter1	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 1) for messages enqueued by this channel
disablesourcespamfilter2	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 2) for messages enqueued by this channel
disablesourcespamfilter3	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 3) for messages enqueued by this channel
disablesourcespamfilter4	(New in 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
disconnectbadburllimit	(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 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
dnsforcetemporary	Treat host not found and no address lookup errors as temporary failures. This option is new in 8.0.1.3.
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_removeroute	Strip source route (@source-route:address) when enqueuing message
envelopetunnel	(New in 8.0.1) Tunnel envelope fields via header lines
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 MS 7.0) Source channel supports Expiry-date: header value
explicitsaslexternal	(New in MS 8.0) Disable automatic use of AUTH EXTERNAL at MAIL FROM
expnallow	(New in MS 6.1) Explicitly enable support of the SMTP command EXPN
expndefault	(New in MS 6.1) Default handling of the SMTP command EXPN
expndisable	(New in MS 6.1) Disable support of the SMTP command EXPN
exproute	Explicit routing for this channel's addresses
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/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; pass along delivery flags, such as trusting a subaddress, IMAP flags, and conversion tags.
forcedreceivedfrom	(New in MS 8.0.1) When constructing a Received: header line, force use of the specified string in the "from ..." clause
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 MS 6.3) Generate a hash of specified message 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	DEPRECATED: Place the message header at the bottom of the message (usage discouraged; use with caution)
headerdecodesrs	Decode SRS encoding of addresses in header lines
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)
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
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
includereceivedip	(New in 8.0.1.2) Include IP address information in the from clauses of generated Received: lines.
inner	Rewrite inner message headers on messages queued to this channel
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
ipbackoff	(New in MS 8.0.2.3) Channel delivery retry backoff intervals for messages in IP backoff mode
ipbackofftimeout	(New in MS 8.0.2.3) Timeout for IP backoff entries made in memcache or Redis.
keepmessagehash	(New in 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 CRLFCRLF 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 MTA message transaction log file
logheader	Include message headers in the MTA message transaction 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
maxconnectionrateperdomain	Implements a limit on the rate of connections made to a domain using the smartsend plugin
maxconnectionsperdomain	Implements limits on the number of simultaneous connections to a domain using the smartsend plugin
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
maxmessagerateperdomain	Implements a limit on the rate of messages sent to a domain using the smartsend plugin
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
mustsaslclient	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)
noaddlineaddrs	Do not attempt the (risky) extraction of recipient addresses from VMS MAIL header lines
noaddresssrs	(New in MS 6.3p1) Addresses matching this channel will not be SRS encoded.
noaddreturnpath	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
noauthhost	(New in 8.0.2.1) Disable use the domain of the authenticated user's primary address to complete addresses
nobangoverpercent	Group A!B%C as (A!B)%C (default)
nobccserver	(New in MS 8.0.2.3) XBCC extension is disabled
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 nodestinationfilter
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 noconvertoctetstream; in legacy configuration, do not convert application/octet-stream material
noconvertoctetstream	(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
nodestinationfilter	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 nsswitch.conf that controls consultation of nameservers
nodnsforcetemporary	Treat host not found and no address lookup errors as permanent failures. This option is new in 8.0.1.3.
nodropblank	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers

noehlo	Never use the SMTP EHLO command
noexpirysource	(New in MS 7.0) Source channel ignores Expiry-date: header value
noexproute	No explicit routing for this channel's addresses
nofileinto	Sieve filter fileinto action has no effect
nofilter	No external-to-LDAP user Sieve filters
noflagtransfer	Disable private XDFLG/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; do not pass along delivery flags, IMAP flags, and conversion tags.
noheaderdecodesrs	Do not decode SRS encoding of addresses in header lines (default)
noheaderread	Do not apply header trimming rules from header 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 on messages queued to this channel
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
nomsexchange	Channel does not serve MS Exchange gateways
nomultigate	Channel does not serve multiple BITNET gateways or LMTP back ends
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
noproxyprotocol	(New in MS 8.1.0.1) Proxy protocol support is disabled
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
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
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
nosourceinner	(New in MS 8.1.0.1) Do not rewrite inner message headers on message sent from this channel
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
notary	Normal NOTARY support
nothurman	Do not perform thurman format conversion
notices	Specify the amount of time which may elapse before notices are sent and messages returned
notick	RESTRICTED: Do not put a ticket in the BSMTP stream
notificationchannel	(New in MS 6.2.) Channel to use when generating notifications
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
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
noturn	Disable the SMTP TURN command
nouma	Do not perform "thurman on demand" format conversion
novrfy	Do not issue SMTP VRFY commands
nowarnpost	Do not send copies of warnings to the postmaster
nox_env_to	Do not add X-Envelope-to: header lines while enqueuing
noxclient	(New in MS 8.0) XCLIENT SMTP extension is disabled
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
passsyntaxerrors	(New in MS 8.0) Disable certain header field syntax error fixup
passthrough	Do no message processing
percentonly	Interpret percent character in an address local-part
percents	Use % routing in the envelope; (legacy configuration's 733 is an alias for `percents`)
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
pool	Specify Job Controller pool in which channel programs run
port	Send to the specified TCP/IP port
postheadbody	Both the message's header and body are copied to the postmaster when a delivery failure occurs
postheadonly	Only the message's header is copied to the postmaster when a delivery failure occurs
presence	RESTRICTED: Not yet implemented
processsecuritymultiparts	Process inside security multiparts
proxyprotocol	(New in MS 8.1.0.1) Proxy protocol support is enabled
randommx	Do MX lookups; randomize returned entries with equal precedence
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
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 *all* 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	(New in MS 8.0.1) RESTRICTED: Disable DSN extension support in SMTP server
rejectsmtplonglines	Reject incoming SMTP messages with illegally long lines
relaxheadertermination	Allow lines containing white space characters to terminate the header of a message
relay	Mark the channel as a relay channel
remotehost	Use remote host's name as the default domain name to complete addresses
reportboth	Generate both header and NOTARY delivery receipt requests from "foreign" delivery receipt requests

reportheader	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
restricted	Apply RFC 1137 restricted encoding to addresses
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
reverse	Apply address reversal to addresses; that is, apply reverse_url LDAP-based address reversal, the reverse database, and the REVERSE mapping to addresses
routelocal	Rewriting should shortcircuit routing addresses
rules	Do channel-specific rewrite rule checks
saslpassauth	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
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)
sendetrn	Send SMTP ETRN command
sendpost	Send copies of failures to the postmaster
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
serviceconversion	(New in MS 7.3) Perform service conversions for messages coming in this channel regardless of CHARSET-CONVERSION
sevenbit	Channel does not support eight bit characters; eight bit characters must be encoded
silentetrn	Honor SMTP client ETRN commands, without echoing channel information
single	Only one envelope To address per message copy
single_sys	Each message copy must be for a single destination system
slave	Channel is serviced only by a slave program
slave_debug	Generate debugging output in the channel's slave program output
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; default on TCP/IP channels
smtp_lf	Accept LF as an SMTP line terminator
sourceblocklimit	Maximum number of MTA blocks allowed per incoming message
sourcebrightmail	DEPRECATED: Alias for sourcespamfilter1
sourcebrightmailoptin	DEPRECATED: Alias for sourcespamfilter1optin
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
sourceconversiontag	(New in MS 7.0.5) Conversion tags to add to message
sourcefilter	Specify the location of the channel's Sieve filter for incoming messages
sourceinner	(New in MS 8.1.0.1) Rewrite inner message headers on message sent from this channel
sourcenosolicit	(New in MS 6.2.) List of solicitation types not accepted
sourcepersonalinc	Leave personal names in incoming message header lines intact
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
sourceroute	Use source routes in the message envelope; (_822 is an alias for sourceroute, synonymous with legacy configuration's 822)

sourcespamfilter	Alias for sourcespamfilter1
sourcespamfilter1	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued by this source channel
sourcespamfilter1optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter2	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued by this source channel
sourcespamfilter2optin	(New in 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 MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued by this source channel
sourcespamfilter3optin	(New in 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 MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued by this source channel
sourcespamfilter4optin	(New in 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
sourcespamfilter5	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued by this source channel
sourcespamfilter5optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter6	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued by this source channel
sourcespamfilter6optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter7	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued by this source channel
sourcespamfilter7optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter8	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued by this source channel
sourcespamfilter8optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilteroptin	Alias for sourcespamfilter1optin
sourcesrs	(New in MS 6.3p1) Messages from this source channel are eligible for SRS encoding
spareN	(New in 8.0.2.2) Fill in the corresponding spare value slot with the specified value
spfhelo	(New in MS 6.3) Perform an SPF lookup at the HELO/EHLO command
spfmailfrom	(New in MS 6.3) Perform an SPF lookup at the MAIL FROM command
spfnone	(New in MS 6.3) Do not perform any SPF lookups
spfrcptto	(New in MS 6.3) Perform an SPF lookup at the RCPT TO command
streaming	Specify degree of protocol streaming for channel to use
subaddressexact	Alias must match exactly, including exact subaddress match
subaddressrelaxed	Alias without subaddress may match
subaddresswild	Alias with subaddress wildcard may match
subdirs	Use multiple subdirectories for messages queued to channel
submit	Mark the channel as a submit-only channel
suppressfinal	Include only original form of address in notification messages
suppressreceivedip	(New in 8.0.1.2) Do not include IP address information in the from clauses of generated Received: lines.
switchchannel	Switch effective source channel to the channel associated with the originating host's source IP
threaddepth	Number of messages triggering new thread with SMTP client
tick	RESTRICTED: Put a ticket in the BSMTP stream
tlsswitchchannel	Switch to specified source channel upon successful TLS negotiation
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
transactionlimit	(New in MS 6.1) Limit the number of transactions per SMTP session (messages per SMTP connection) accepted

truncatesmtplonglines	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
uma	Perform thurman conversion if would be MIME-ifying the message anyway for other reasons
unrestricted	Do not apply RFC 1137 restricted encoding to addresses
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
urgentnotices	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority
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
usepermanenterror	(New in MS 8.0) Override use_permanent_error MTA option on a per-source-channel basis
user	DEPRECATED as of MS 8.0; instead use the pipeuser option in restricted.cnf. Specify account under which to run the pipe channel
usereplyto	Specify mapping of Reply-to: header
useresent	Specify mapping of Resent- headers for non RFC 822 environments
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
usetemporaryerror	(New in MS 8.0) Override use_temporary_error MTA option on a per-source-channel basis
utf8header	(New in MS 8.0.2) Enable unconditional EAI support
utf8negotiate	(New in MS 8.0.2) Enable EAI support
utf8strict	(New in MS 8.0.2) Enable EAI support with strict enforcement
uucp	Alias for bangstyle; (use UUCP ! routing in the envelope)
validatelocalnone	Enqueuing channels perform no validation check on the local part of addresses they enqueue to this channel
validatelocalsystem	Enqueuing channels check that the local part of addresses they enqueue to this channel matches an account on the system
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
viaaliasoptional	Alias match not required
viaaliasrequired	Alias match required
vrfyallow	Provide informative responses to SMTP VRFY command
vrfydefault	Default responses to SMTP VRFY command, according to channel's TCP/IP-channel-specific HIDE_VERIFY option setting
vrfyhide	Provide obfuscatory responses to SMTP VRFY command
warnpost	Send copies of warnings to the postmaster
wrapsmtplonglines	Wrap incoming SMTP messages with illegally long lines
x_env_to	Add X-Envelope-to: header lines while enqueuing
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

46.3.2 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:

- Addresses
- Attachments and MIME processing
- BSMTP-specific
- Character sets and eight bit data
- Conversion tag and service conversion
- Display labels
- DKIM
- Error interpretation
- File creation in the MTA queue area
- Gateway/firewall/mailhub/Message Router channel connection
- Headers
- Host names
- Incoming channel matching and switching
- Logging and debugging
- Long address lists or headers
- Message hash
- Message tracking
- MLS
- Notification messages and postmaster messages
- Processing control and job submission
- Sensitivity limits
- Sieve filters and delivery flags
- Size limits on messages
- Spam/virus filter package use
- SMTP and LMTP protocol
- TCP/IP connections, DNS lookups, and SOCKS connections
- TLS and SASL

Channel options shown in **bold face type** are defaults; channel options marked with + are only supported under OpenVMS.

Table 46.4 Channel options grouped by functionality

Option	Usage
Addresses	
_733	Use % routing in the envelope; alias for percents
_822	Use source routes in the envelope; alias for sourceroute
acceptalladdresses	(New in MS 6.1) Accept messages despite certain errors that would normally cause message rejection
accepttemporaryfailures	(New in 8.0.1.1.0) Accept messages despite recipient address temporary error conditions (*e.g.*, over quota, LDAP server unavailable, Spam/virus filter unavailable, *etc.*).
acceptvalidaddresses	(New in MS 6.1) Perform normal rejection checks on incoming messages
addlineaddrs	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
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
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; *i.e.*, query alias file and alias database and alias options and perform alias_url* lookups
aliasmagic	(New in 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 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
aliaswild	Do an * lookup if no exact alias match is found
authhost	(New in 8.0.2.1) Use the domain of the authenticated user's primary address to complete addresses
authrewrite	Use SMTP AUTH information in header
bangonly	Source channel: disable interpretation of % host-routing
bangoverpercent	Group A!B%C as A!(B%C)
bangstyle	Use UUCP ! routing in the envelope; (uucp from legacy configuration is an alias for bangstyle)
defaulthost	Specify a domain name to use to complete addresses
defertemporaryfailures	(New in 8.0.1.1.0) Defer, with SMTP temporary rejection, recipient addresses with temporary error conditions (*e.g.*, over quota, LDAP server unavailable, Spam/virus filter unavailable, *etc.*).
dequeue_removeroute	Alias for dequeueremoveroute; in legacy configuration, strip source route (@source-route:address) when dequeuing message
dequeueremoveroute	(Unified Configuration only) Strip source route (@source-route:address) when dequeuing message
destinationsrs	(New in MS 6.3p1) Messages destined out this channel are eligible for SRS encoding
enqueue_removeroute	Alias for enqueueremoveroute; in legacy configuration, strip source route (@source-route:address) when enqueuing message
enqueueremoveroute	(Unified Configuration only) Strip source route (@source-route:address) when enqueuing message
exproute	Explicit routing for this channel's addresses
headerdecodesrs	Decode SRS encoding of addresses in header lines
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
noaddlineaddrs	Do not attempt the (risky) extraction of recipient addresses from VMS MAIL header lines
noaddresssrs	(New in MS 6.3p1) Addresses matching this channel will not be SRS encoded.
noaddreturnpath	Do not add a Return-Path: header line
noaddrtypescan	Do not store recipient address "type" in an envelope flag
noauthhost	(New in 8.0.2.1) Disable use the domain of the authenticated user's primary address to complete addresses
nobangoverpercent	Group A!B%C as (A!B)%C (default)
nodefaulthost	Do not specify a domain name to use to complete addresses
nodestinationsrs	(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
noheaderdecodesrs	Do not decode SRS encoding of addresses in header lines (default)
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 *all* 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.e.*, 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
spareN	(New in 8.0.2.2) Fill in the corresponding spare value slot with the specified value
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
utf8header	(New in MS 8.0.2) Enable unconditional EAI support
utf8negotiate	(New in MS 8.0.2) Enable EAI support
utf8strict	(New in MS 8.0.2) Enable EAI support with strict enforcement
validatelocalnone	Enqueuing channels perform no validation check on the local part of addresses they enqueue to this channel
validatelocalsystem	Enqueuing 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	
conditionalsecuritymultiparts	Process inside security multiparts, 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
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
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
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
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 and EAI	
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
utf8header	(New in MS 8.0.2) Enable unconditional EAI support
utf8negotiate	(New in MS 8.0.2) Enable EAI support
utf8strict	(New in MS 8.0.2) Enable EAI support with strict enforcement
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
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 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	
addrsperfile	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 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
Headers	
addreturnpath	Add a Return-Path: header line
authhost	(New in 8.0.2.1) Use the domain of the authenticated user's primary address to complete addresses
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
envelopetunnel	(New in Cayenne) Tunnel envelope fields via header lines
fixsyntaxerrors	(New in MS 8.0) Correct syntax errors in header fields
forcedreceivedfrom	(New in MS 8.0.1) When constructing a Received: header line, force use of the specifed string in the "from ..." clause
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)
headerdecodesrs	Decode SRS encoding of addresses in header lines
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.e.*, 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
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)
includereceivedip	(New in 8.0.1.2) Include IP address information in the from clauses of generated Received: lines.
inner	Rewrite inner message headers on message queued to this channel
innertrim	Apply header trimming rules from a header option file to inner message headers (use with caution)
limitheadertermination	(New in MS 7.0.5) Only CRLFCRLF terminates message header
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

missingrecipientpolicy	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
noaddreturnpath	Do not add a Return-Path: header line
noauthhost	(New in 8.0.2.1) Disable use the domain of the authenticated user's primary address to complete addresses
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
noheaderdecodesrs	Do not decode SRS encoding of addresses in header lines (default)
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 on messages queued to this channel
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
nosourceinner	(New in MS 8.1.0.1) Do not rewrite inner message headers on message sent from this channel
nox_env_to	Do not add X-Envelope-to: header lines while enqueuing
passsyntaxerrors	(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
sourceinner	(New in MS 8.1.0.1) Rewrite inner message headers on message sent from this channel
sourcepersonalinc	Leave personal names in incoming message header lines intact
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
suppressreceivedip	(New in 8.0.1.2) Do not include IP address information in the from clauses of generated Received: 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 enqueuing
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 mailSMTPSubmitChannel LDAP attribute (or as of MS 8.0, whatever LDAP attribute is named by the ldap_auth_attr_submit_channel MTA option
switchchannel	Switch effective source channel to the channel associated with the originating host's source IP
tlsswitchchannel	Switch to specified source 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	
alternatechannel	Messages that exceed a channel's alternateblocklimit, alternatelinelimit, or alternaterecipientlimit will be diverted to the channel's specified alternatechannel
alternaterecipientlimit	Divert messages that exceed the specified number of recipients to the alternatechannel
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 *all* 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 MS 7.0) Source channel supports Expiry-date: header value
includefinal	Include final form of address in delivery notifications
language	Specify a preferred language for messages that have none
noexpirysource	(New in MS 7.0) Source channel ignores Expiry-date: header value
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
reportboth	Generate both header and NOTARY delivery receipt requests from "foreign" delivery receipt requests
reportheader	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
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
ipbackoff	(New in MS 8.0.2.3) Channel delivery retry backoff intervals for messages in IP backoff mode
ipbackofftimeout	(New in MS 8.0.2.3) Timeout for IP backoff entries made in memcache or Redis.
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 pecify 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 inMS 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
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
pool	Specify Job Controller pool in which channel programs run
secondclassafter	Specify time delay before channel runs for secondclass priority messages
secondclassblocklimit	Force messages above this size to third class priority
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
user	DEPRECATED as of MS 8.0; instead use the pipeuser option in restricted.cnf. Specify account under which to run the pipe channel
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/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; pass along delivery flags, such as trusting a subaddress, IMAP flags, and conversion tags.
noaddrtypescan	(New in MS 7.0.5) Do not store recipient address "type" in an envelope flag

nochannelfilter	Alias for nodestinationfilter
nodestinationfilter	Do not perform channel filtering on outgoing messages; (nochannelfilter is an alias for nodestinationfilter)
nofileinto	Sieve "fileinto" action has no effect
nofilter	No external-to-LDAP user Sieve filters
noflagtransfer	Disable private XDFLG/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; do not pass along delivery flags, IMAP flags, and conversion tags.
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 the channel's Sieve filter 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
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
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
Spam/virus filter package use	
destinationbrightmail	DEPRECATED: Alias for destinationspamfilter1
destinationbrightmailoptin	DEPRECATED: Alias for destinationspamfilter1optin
destinationspamfilter	Alias for destinationspamfilter1
destinationspamfilter1	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued to this destination channel
destinationspamfilter1optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter2	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued to this destination channel
destinationspamfilter2optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter3	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued to this destination channel
destinationspamfilter3optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter4	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued to this destination channel
destinationspamfilter4optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter5	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued to this destination channel
destinationspamfilter5optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter6	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued to this destination channel
destinationspamfilter6optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter7	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued to this destination channel
destinationspamfilter7optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilter8	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued to this destination channel
destinationspamfilter8optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued to this destination channel, with the optin value provided as argument
destinationspamfilteroptin	Alias for destinationspamfilter1optin
disabledestinationbrightmail	DEPRECATED: Alias for disabledestinationspamfilter1
disabledestinationspamfilter	Alias for disabledestinationspamfilter1
disabledestinationspamfilter1	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 1) for messages enqueued to this channel
disabledestinationspamfilter2	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 2) for messages enqueued to this channel

disabledestinationspamfilter3	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 3) for messages enqueued to this channel
disabledestinationspamfilter4	(New in MS 6.2) Disable spam/virus filtering (via spam/filter package 4) for messages enqueued to this channel
disabledestinationspamfilter5	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/filter package 5) for messages enqueued to this channel
disabledestinationspamfilter6	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/filter package 6) for messages enqueued to this channel
disabledestinationspamfilter7	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/filter package 7) for messages enqueued to this channel
disabledestinationspamfilter8	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/filter package 8) for messages enqueued to this channel
disablesourcebrightmail	DEPRECATED: Alias for disablesourcespamfilter1
disablesourcespamfilter	DEPRECATED: Alias for disablesourcespamfilter1
disablesourcespamfilter1	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 1) for messages enqueued by this channel
disablesourcespamfilter2	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 2) for messages enqueued by this channel
disablesourcespamfilter3	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 3) for messages enqueued by this channel
disablesourcespamfilter4	(New in MS 6.2) Disable spam/virus filtering (via spam/virus filter package 4) for messages enqueued by this channel
disablesourcespamfilter5	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/virus filter package 5) for messages enqueued by this channel
disablesourcespamfilter6	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/virus filter package 6) for messages enqueued by this channel
disablesourcespamfilter7	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/virus filter package 7) for messages enqueued by this channel
disablesourcespamfilter8	(New in MS 6.3-0.15) Disable spam/virus filtering (via spam/virus filter package 8) for messages enqueued by this channel
sourcebrightmail	DEPRECATED: Alias for sourcespamfilter1
sourcebrightmailoptin	DEPRECATED: Alias for sourcespamfilter1optin
sourcespamfilter	Alias for sourcespamfilter1
sourcespamfilter1	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued by this source channel
sourcespamfilter1optin	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 1) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter2	(New in MS 6.2) Enable spam/virus filtering (by spam/virus filter package 2) for messages enqueued by this source channel
sourcespamfilter2optin	(New in 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 MS 6.2) Enable spam/virus filtering (by spam/virus filter package 3) for messages enqueued by this source channel
sourcespamfilter3optin	(New in 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 MS 6.2) Enable spam/virus filtering (by spam/virus filter package 4) for messages enqueued by this source channel
sourcespamfilter4optin	(New in 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
sourcespamfilter5	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued by this source channel
sourcespamfilter5optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 5) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter6	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued by this source channel
sourcespamfilter6optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 6) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter7	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued by this source channel
sourcespamfilter7optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 7) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilter8	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued by this source channel
sourcespamfilter8optin	(New in MS 6.3) Enable spam/virus filtering (by spam/virus filter package 8) for messages enqueued by this source channel, with the optin value provided as argument
sourcespamfilteroptin	Alias for sourcespamfilter1optin
SMTP and LMTP protocol	
allowetrn	Honor SMTP client ETRN commands
bccserver	(New in MS 8.0.2.3) XBCC extension is enabled
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
destinationnosolicit	(New in MS 6.2) List of solicitation types not accepted

disableetrn	Disable support for the ETRN SMTP command
disconnectbadburllimit	(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 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 MS 6.1) Explicitly enable support of the SMTP command EXPN
expndefault	(New in MS 6.1) Default handling of the SMTP command EXPN
expndisable	(New in MS 6.1) Disable support of the SMTP command EXPN
flagtransfer	Support private XDFLG/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; pass along delivery flags, such as trusting a subaddress, IMAP flags, and conversion tags.
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
maxconnectionrateperdomain	Implements a limit on the rate of connections made to a domain using the smartsend plugin
maxconnectionsperdomain	Implements limits on the number of simultaneous connections to a domain using the smartsend plugin
maxmessagerateperdomain	Implements a limit on the rate of messages sent to a domain using the smartsend plugin
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.
nobccserver	(New in MS 8.0.2.3) XBCC extension is disabled
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
noflagtransfer	Disable private XDFLG/XAFLG SMTP/LMTP extensions and (new in MS 8.0.2.3) XCONVTAG SMTP extension; do not pass along delivery flags, IMAP flags, and conversion tags.
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
noproxyprotocol	(New in MS 8.1.0.1) Proxy protocol support is disabled
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
proxyprotocol	(New in MS 8.1.0.1) Proxy protocol support is enabled
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 *all* 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	(New in MS 8.0.1) RESTRICTED: Disable DSN extension support in SMTP server
rejectsmtplonglines	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; default on TCP/IP channels
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 MS 6.1) Limit the number of transactions per SMTP session (messages per SMTP connection) accepted
truncatesmtplonglines	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 TCP/IP-channel-specific HIDE_VERIFY option setting
vrfyhide	Provide obfuscatory responses to SMTP VRFY command
wrapsmtplonglines	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
connectalias	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
dnsforcetemporary	Treat host not found and no address lookup errors as temporary failures. This option is new in 8.0.1.3.
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 *
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 `nsswitch.conf` that controls consultation of nameservers
nodnsforcetemporary	Treat host not found and no address lookup errors as permanent failures. This option is new in 8.0.1.3.
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
spfhelo	(New in MS 6.3) Perform an SPF lookup at the HELO/EHLO command
spfmailfrom	(New in MS 6.3) Perform an SPF lookup at the MAIL FROM command
spfnone	(New in MS 6.3) Do not perform any SPF lookups
spfrcptto	(New in 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
explicitsaslexternal	(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
implicitsaslexternal	(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
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
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
saslpassauth	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.

46.3.3 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.

46.3.3.1 _733 Option

The legacy configuration 733 channel option has been replaced in Unified Configuration by percents.

46.3.3.2 _822 Option

The legacy configuration 822 channel option has been replaced in Unified Configuration by sourceroute.

46.3.3.3 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. With acceptalladdresses, various sorts of normally "invalid: recipient addresses will be accepted: syntactically invalid addresses, addresses of local users with problematic status (such as being "over quota" or "disabled" -- see the ldap_user_mail_status, ldap_group_mail_status, and ldap_domain_attr_mail_status MTA options), and recipients of messages that fail message acceptability checks. Relevant message acceptability checks include checking on whether a message: exceeds a configured sensitivity limit, or contains unnegotiated eight bit data (on a source channel marked eightstrict or utf8strict), or contains too-long-for-SMTP lines (on a source channel marked rejectsmtplonglines), or exceeds a configured message size limit (such as blocklimit, sourceblocklimit, block_limit, ldap_blocklimit, ldap_sourceblocklimit, ldap_maximum_message_size, ldap_domain_attr_blocklimit, ldap_domain_attr_sourceblocklimit, alias_blocklimit, or [BLOCKLIMIT]) or message line limit (such as linelimit, line_limit, alias_linelimit, or [LINELIMIT]) when the excessive size is detected after the DATA stage, or is entirely lacking in recipient header lines (on a source channel marked missingrecipientpolicy 6, or if the MTA option missing_recipient_policy=6 is set), or is rejected via the AUTH_REWRITE mapping table. Setting acceptalladdresses will also disable the SMTP protocol level rejection feature of the Sieve "refuse" action; with acceptalladdresses, addresses to be rejected due to application of a "reject" action will be accepted during the SMTP dialogue, and instead a subsequent non-delivery DSN generated.

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 during the SMTP dialogue. `acceptvalidaddresses` is the default.

Note that the MTA's default, `acceptvalidaddresses`, behavior is normally far preferable to `acceptalladdresses`: rejecting invalid addresses and messages "up front" rather than accepting them initially only to have to generate and send back a non-delivery notification later has advantages that should hardly need to be pointed out, including:

- Less work for the MTA. The reductions in work are in several areas, including: less processing required on incoming messages (in cases where a recipient address can be detected as invalid), less work generating non-delivery notification messages, less work attempting to deliver non-delivery notification messages, *etc.*
- Less network bandwidth required.
- Due to the above two points, less potential impact on throughput of other, valid, messages.
- Immediate (via an SMTP error) notification to e-mail clients of the error in their message, rather than a delayed notification via DSN.
- Avoiding "blow-back" spam, in cases of so-called "joe-job" (forged envelope From) spam.

However, `acceptalladdresses` can sometimes be useful in certain limited cases, such as when attempting to support poorly designed e-mail clients that fail to report SMTP errors correctly to the user. In such cases a DSN, while providing only a delayed indication rather than an immediate indication of an address or message problem, may be the only way to convey the MTA's accurate error information to the user. So appropriate use of `acceptalladdresses` tends to be restricted to cases of special incoming TCP/IP channels dedicated for submission of messages by applications or user e-mail clients with known limitations; `*switchchannel` effects, or else channels associated with special Dispatcher services listening on special ports or interface addresses, are features typically used for setting up such special channels.

Unconditional address acceptance (`acceptalladdresses`) 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 MS 6.1.

Compare with the (new in 8.0.1.1.0) `accepttemporaryfailures` and `defertemporaryfailures` channel options.

46.3.3.4 Envelope recipient error handling (`accepttemporaryfailures`, `defertemporaryfailures`)

(New in MS 8.0.1.1.0.) The `accepttemporaryfailures` and `defertemporaryfailures` source channel options control how recipient address temporary errors such as user over quota, spam filter temporarily unavailable, LDAP server unavailable, and so on are handled by the SMTP server. `defertemporaryfailures` is the default, and causes such errors to return an immediate 4yz error in the SMTP session. Setting `accepttemporaryfailures` causes

the address to be accepted, but the message will be placed in the reprocess queue and retried until it either times out and is returned or the error clears.

These channel options are only intended for use on SMTP channels. Placing them on an internal channel may have unexpected results.

Compare with the `acceptalladdresses` and `acceptvalidaddresses` channel options.

46.3.3.5 Filling in missing header addresses (`addlineaddrs`, `noaddlineaddrs`)

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 `addlineaddrs` option is placed on a source channel it instructs submission agents to use whatever additional information is available to reconstruct the missing information. `noaddlineaddrs` is the default and prevents this from happening.

Currrently the only submission agent that supports `addlineaddrs` is the VMS MAIL interface in PMDF. This option has no effect on any Messaging Server channel.

46.3.3.6 Controlling Sender Rewriting Scheme (SRS) rewriting (`addresssrs`, `noaddresssrs`, `destinationsrs`, `nodestinationsrs`, `sourcesrs`, `nosourcesrs`)

(New in 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 `destinationsrs` option, and (3) Rewriting the MAIL FROM address must have matched a channel marked with the `addresssrs` option. `noaddresssrs`, `nodestinationsrs` and `nosourcesrs` are the default for all channels.

See also the `headerdecodesrs` and `noheaderdecodesrs` channel options.

46.3.3.7 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 informtion, the MTA's delivery flag transfer facilities may be used to transport this information between MTAs; see the deliveryflags channel option.

46.3.3.8 Force "detour" routing of hosted users (aliasdetourhost, aliasoptindetourhost)

The (new in iMS 5.2p2, and 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 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.

One disadvantage of using `aliasoptindetourhost` is that all alias expansion is deferred, including expansions that result in mail being discarded. This can lead to messages sent to the bitbucket wasting processing resources.

One way to work around this problem is to use a `$*` rewrite rule and an associated mapping to direct such addresses to the bitbucket channel, bypassing any use of aliasdetourhost. For example:

```
$*              $E$F${bitbucket_check,$U$@$H}


BITBUCKET_CHECK

  noreply@example.com     $Y$$U%$$H@bitbucket-daemon
  unattended@example.net  $Y$$U%$$H@bitbucket-daemon
```

This will caused any mail sent to noreply@example.com and unattended@example.net to be discarded before any other lookups or redirection.

46.3.3.9 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.

46.3.3.10 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 46.5 Alias_magic digits

Value	Meaning
0	No operation
1	Personal alias database
2	Local name table aliases
3	System alias database
4	System alias file
5	Special aliases

6	alias_url0
7	alias_url1
8	alias_url2
9	alias_url3

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.

46.3.3.11 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.

46.3.3.12 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 or new in MS 8.0 the value of whatever attribute is named by the ldap_auth_attr_sender MTA option, 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 46.6 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: *src-chan|env-from|[resent-]sender|[resent-]from|auth-originator*
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; that is, when this bit is set then probes take the form *env-from|[resent-]sender|[resent-]from|auth-originator|auth-param*
9	512	(New in MS 7.0.5) If set, the final tag set via $T in the *_ACCESS mappings will be prefixed to the AUTH_REWRITE mapping probe; that is, when this bit is set then probes take the form: *ACCESS-tag|env-from|[resent-]sender|[resent-]from|auth-originator*

46.3.3.13 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` channe; option forces the former A!(B%C) interpretation. `nobangoverpercent` forces the latter (A!B)%C interpretation. `nobangoverpercent` is the default for all channels.

46.3.3.14 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.

46.3.3.15 Recipient validity date check (`checkrrvs`, `ignorerrvs`)

The `checkrrvs` and `ignorerrvs` source channel options, for support of RFC 7293, are new in MS 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.

46.3.3.16 Clone messages to alternate destination (`clonehosts`)

(New in MS 8.0) 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
```

46.3.3.17 Host name to use when correcting incomplete addresses (`authhost`, `noauthhost`,`defaulthost`, `nodefaulthost`, `remotehost`, `noremotehost`)

The MTA often receives addresses consisting of a bare local-part and no "@domain" from misconfigured or incompliant mailers and SMTP clients. (Note that the standards do in fact require acceptance of one special address - "postmaster" in SMTP, but do not state how such an address is to be represented in a message header.) This happens often enough that simply disallowing such addresses is rarely an acceptable strategy.

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", producing "mrochek@acme.com").

The set of options described here control how this domain name is selected. In this process the MTA makes a distinction between envelope To (RCPT TO) addresses versus addresses appearing in all other contexts (header, MAIL FROM), as well as distinguishing SUBMIT from SMTP.

In the following table showing what domain is used when various options or protocols are used, *o.org* is the domain attached to the local channel, *a.com* is the domain associated with the current authenticated user's primary email address, *l.com* is the first argument of the `defaulthost` option setting on the current source channel, *r.com* is the second argument of the `defaulthost` option setting on the current source channel, and *r.edu* is the domain associated with the remote SUBMIT/SMTP client.

Table 46.7 Options controlling missing domain fixups

Option	Protocol	Header/ MAIL FROM	RCPT TO
`authhost` (and authenticated)	SUBMIT/SMTP	@a.com	@a.com
`defaulthost`	SUBMIT/SMTP	@o.org	@o.org
`defaulthost` *l.com*	SUBMIT	@l.com	@l.com
`defaulthost` *l.com*	SMTP	@l.com	@o.org
`defaulthost` *l.com r.com*	SUBMIT	@l.com	@l.com
`defaulthost` *l.com r.com*	SMTP	@l.com	@r.com
`nodefaulthost` (MS 8.0.2.1 or later)	SUBMIT/SMTP	<no fixup>	<no fixup>
`remotehost`	SUBMIT	@r.edu	@r.edu
`remotethost`	SMTP	@r.edu	@o.org
<no options>	SUBMIT/SMTP	@o.org	@o.org

Note: The option in the preceding table are shown in precedence order, that is, a given option setting will override option settings further down the table.

The usual configuration is to have both arguments to `defaulthost` set to the `defaultdomain` on the defaults channel, overriding any use of the local channel host. It may be appropriate in multitenant configurations to set `authhost` on all channels that are marked with the `submit` channel option

Use of the `remotehost` channel option may be considered in rare cases where remote SMTP client exist in another administrative domain which partial addresses to refer to their own users.

Note that rewrite rules can make use of whatever default has been selected via the the `$G` and `$`*`n`*`G` substitutions.

Note that the `switchchannel`, `saslswitchchannel`, `tlsswitchchannel` and various other options can be used to associated incoming SMTP connections with a particular channel, and thus control what set of options are used.

46.3.3.18 `dequeue_removeroute` Option

The legacy configuration `dequeue_removeroute` channel option has been replaced in Unified Configuration by `dequeueremoveroute`.

46.3.3.19 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 `dequeueremoveroute` 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 `enqueueremoveroute` 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 `enqueueremoveroute` 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_removeroute` and `enqueue_removeroute` options are aliases for `dequeueremoveroute` and `enqueueremoveroute`, respectively. These obsolete options are not supported in unified configuration mode.

46.3.3.20 enqueue_removeroute Option

The legacy configuration `enqueue_removeroute` channel option has been replaced in Unified Configuration by `enqueueremoveroute`.

46.3.3.21 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 repliable, 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 (`channel:l.official_host_name`) (or the current alias for the local system, the current channel's `local_host_alias`) 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.

46.3.3.22 Controlling Sender Rewriting Scheme (SRS) rewriting in header lines (`headerdecodesrs`, `noheaderdecodesrs`)

New in 8.0.1.3. The `headerdecodesrs` channel option, if set on the current source or destination channel, causes any SRS-encoded addresses found in any address header fields to be decoded. The decoding in this case does not enforce password or timeout checks; it is sufficient that the SRS domain match for decoding to occur.

The default is `noheaderdecodesrs`.

46.3.3.23 Local-channel-like behavior (`localbehavior`, `nolocalbehavior`)

Use of `localbehavior` or `nolocalbehavior` 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 `nolocalbehavior` is not usually appropriate; instead, channel defaults, optionally modified in specific ways via options such as `aliaslocal` or `routelocal`, should be used.

46.3.3.24 Handling messages that lack any recipient headers (`missingrecipientpolicy`)

RFC 822, the original Internet message format standard, had a requirement that all messages contain at least one recipient header field: a To:, Cc:, or Bcc:.

As of RFC 2822, the original update to RFC 822, relaxed the RFC 822 requirement and allowed submitted messages to be lacking in any recipient header line. This change was carried forward to the current message format standard, RFC 5322.

However, there are still MTAs around that operate according to RFC 822, and in particular may try to be helpful by adding a To: field containing all of the envelope recipients when no recipient fields are present. As such, it may be 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 provides various capabilities that may be useful in addressing this issue. It takes an integer value specifying what approach to use to "fix" messages with no recipient field; 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 MS 6.2 is equivalent to a value of 1 meaning that messages are passed through unchanged---in MS 6.0 and 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 46.8 `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 5322) 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.e.*, ;) To: header line. The phrase used in the group construct is controlled by the `missing_recipient_group_text` 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 `acceptalladdresses` channel option, if used, modifies the timing and form of the rejection.)

Note that the `missing_recipient_policy` MTA option can be used to set an MTA system default for this behavior.

46.3.3.25 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.)

46.3.3.26 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.

46.3.3.27 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.

46.3.3.28 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.

46.3.3.29 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 option, 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 option 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.

46.3.3.30 Extra value channel options: `spare*` (string)

The `spare`*`N`* source channel options provide functionality (*`N`* = 1,...,18) are analogous to the attributes named by the `ldap_spare_`*`N`* MTA options, except with these options the corresponding spare slot is filled in when the source channel is selected.

The spare value slots are 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.*).

46.3.3.31 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 MS 6.1, the subaddress support in aliases includes `alias_urlN` alias lookups; that is, as of 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.

46.3.3.32 Alias and address channel options: `usereversedatabase` (bitmask)

As of MS 8.0.1.3, the `usereversedatabase` source channel option provides a way to override the setting of the `use_reverse_database` MTA option on a channel by channel basis. All bits have identical meaning in both options.

46.3.3.33 uucp Option

The legacy configuration `uucp` channel option has been replaced in Unified Configuration by `bangstyle`.

46.3.3.34 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 enqueuing 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 enqueuing 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 enqueuing 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`.

46.3.3.35 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 enqueing 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.

46.3.4 Attachments and MIME processing channel options

A number of channel options affect the processing of so-called attachments and MIME parts. (Note that these are channel options which can generally be set on *any*type of channel. See also Message conversions for optional, more complex, processing of attachments and MIME parts, sometimes involving a "hop" through the specialized Conversion channel.)

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`, `noinner`, `sourceinner`, and `nosourceinner` 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.

46.3.4.1 Processing within security multiparts (conditionalsecuritymultiparts, processsecuritymultiparts, retainsecuritymultiparts)

The conditionalsecuritymultiparts, processsecuritymultiparts, 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 processsecuritymultiparts, 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 processsecuritymultiparts, but retaining any "preamble" material inside the multipart.

46.3.4.2 convert_octet_stream Option

The legacy configuration convert_octet_stream channel option has been replaced in Unified Configuration by convertoctetstream.

46.3.4.3 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.

46.3.4.4 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.

46.3.4.5 Encoding header interpretation (`ignorencoding`, `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 in 6.3 `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.

46.3.4.6 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 `*smtplonglines` 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.

46.3.4.7 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*.

46.3.4.8 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 accomodate 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.

46.3.4.9 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.

46.3.4.10 `noconvert_octet_stream` Option

The legacy configuration `noconvert_octet_stream` channel option has been replaced in Unified Configuration by `noconvertoctetstream`.

46.3.4.11 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.

46.3.4.12 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.

46.3.5 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.

46.3.5.1 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 remainer 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.

46.3.5.2 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.

46.3.5.3 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.

46.3.6 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.

46.3.6.1 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).

46.3.6.2 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. (Note that the timing of the rejection can be postponed via the `acceptalladdresses` channel option.)

The MS 8.0.2 release adds MTA support for EAI messages and the SMTPUTF8 extension. EAI messaging is documented in RFC 6530 (overview), RFC 6531 (SMTPUTF8 extension), RFC 6532 (header format changes), and RFC 6533 (DSN and MDN format changes). Three additional channel options have been added to enable and control EAI support:

`utf8header` As a SMTP source channel option, offer the SMTPUTF8 SMTP extension. As a destination channel option, allow enqueue and dequeue of EAI messages unconditionally; in particular, the SMTPUTF8 extension will not be required. Note that delivery of EAI messages via SMTP/LMTP to a non-EAI system is a standards violation.

`utf8negotiate` As a SMTP source channel option, offer the SMTPUTF8 SMTP extension. As a destination channel option, allow enqueue unconditionally. On

dequeue, require the SMTPUTF8 extension be offered by SMTP/LMTP servers for EAI messages with EAI recipient (RCPT TO) addresses or un-downgradable EAI originator (MAIL FROM) addresses.

`utf8strict` Same as `utf8negotiate` as far as SMTP/LMTP destination channels are concerned. But additionally, for SMTP/LMTP servers (source channels), reject messages that contain 8bit headers without having negotiated the SMTPUTF8 extension and 8bit bodies without having negotiated the 8BITMIME extension. (Note that such rejections are postponed if the `acceptalladdress` channel option is used.)

As of MS 8.0.2.2 the channel default for internal channels has changed from `eightnegotiate` to `utf8always`. Note, however, that the channel default remains `eightnegotiate` on all other channels; EAI support must therefore be explicitly enabled. A channel set to `eightnegotiate` will not offer the SMTPUTF8 extensions or allow EAI messages with EAI recipient or originator addresses to be enqueued. Also note that the Message Store does not support EAI at the present time.

46.3.6.3 Unencoded non-ASCII headers (`headerset7`, `headerset8`, `headersetesc`)

In extremely rare situations very old and standards-incompliant SMTP servers must be accomodated. 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.

46.3.6.4 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="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.

46.3.7 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.

46.3.7.1 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.

46.3.7.2 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.

46.3.8 Display label channel options

A couple of channel options are intended for labelling purposes, for future use.

46.3.8.1 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.

46.3.9 DKIM channel options

As of MS 7.0.5, a number of channel options can be used to affect message handling based on the presence of DKIM signatures. See also the DKIM MTA options.

MS 8.0.2.3 introduced native DKIM signing capbilities, controled by various DKIM channel options.

46.3.9.1 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.

46.3.9.2 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.

46.3.9.3 Channel-based DKIM signing (`destinationdkimidentityN`, `destinationdkimselectorN`, `sourcedkimidentityN`, `sourcedkimselectorN`)

The `destinationdkimidentityN/destinationdkimselectorN` and `sourcedkimidentityN/sourcedkimselectorN` channel options provide the ability to apply DKIM signatures to messages based on destination and source channels, respectively.

The specification of one or more DKIM identities enables signing as enqueued messages are written to disk. Each identity is used to access a corresponding private key in PEM format stored as DATAROOT/dkim_private/<identity>/<selector>.pem.

Multiple "slots" are provided so that multiple signatures can be applied simultaneously. Slot values starting at 0 appear at the end of the option name. At present four slots are available, so N can range from 0 to 3.

Identity values specifies the DKIM identity. This can take the form "user@domain", "@domain", "domain", or the special value "*" can be given which specifies that the domain of From: header address should be used.

Some DKIM errors will be sent to syslog (see `mta.sndopr_priority`). Additional diagnostics are available by turning on MTA debugging, setting `mta.mm_debug` to at least 3, and setting dkim-related `base.debugkeys` as desired.

New in MS 8.1, when a "*" is specified an additional check is made to see if a `DKIM_SIGN_DOMAINS` mapping exists. If it does it is consulted with a probe of the form:

`source-channel|destination-channel|from-address`

If $Y is specified the result of the mapping is used as the DKIM identity. If $N is specified DKIM signing is disabled.

For example, a mapping that would cause the foo.example.com subdomain to be signed with the bar.example.com identity, all other subdomains of example.com to be signed with the example.com identity, and example.com without a subdomain not to be signed would look like:

```
DKIM_SIGN_DOMAINS

  *|*|*@foo.example.com   bar.example.com$Y
  *|*|*@*.example.com     example.com$Y
  *|*|*@example.com       $N
```

The optional selector value given by the `destinationdkimselectorN` options specify a space-separated list of selectors that may be used. The newest key associated with the identity will be used if no selector is specified.

Note that `CONVERSION` mappings can also specifify DKIM identity and selector values. If multiple values for the same slot are specified the `CONVERSION` mapping value will override any destination channel value, which in turn will override any source channel value.

46.3.10 Error interpretation channel options

A couple of channel options override general MTA option settings regarding error interpretation.

46.3.10.1 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.

46.3.11 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.

46.3.11.1 Addresses per message copy (`multiple`, `addrsperfile`, `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`, `addrsperfile`, `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 `addrsperfile` 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 accomodate them.

Note that the default of `addrsperfile` depends on the channel type. For most channel types, `addrsperfile` defaults to effectively unlimited (2147483647), but setting an `smtp*` option on a channel causes `addrsperfile` to default to 99, while setting an `lmtp*` option on a channel causes `addrsperfile` 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`.

46.3.11.2 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 on a per-source-channel basis 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 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 earler 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.

46.3.11.3 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.

46.3.12 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.

46.3.12.1 Force "detour" routing of hosted users (`aliasdetourhost`, `aliasoptindetourhost`)

The (new in iMS 5.2p2, and 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 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.

One disadvantage of using `aliasoptindetourhost` is that all alias expansion is deferred, including expansions that result in mail being discarded. This can lead to messages sent to the bitbucket wasting processing resources.

One way to work around this problem is to use a `$*` rewrite rule and an associated mapping to direct such addresses to the bitbucket channel, bypassing any use of aliasdetourhost. For example:

```
$*              $E$F${bitbucket_check,$U$@$H}


BITBUCKET_CHECK

  noreply@example.com     $Y$$U%$$H@bitbucket-daemon
  unattended@example.net  $Y$$U%$$H@bitbucket-daemon
```

This will caused any mail sent to noreply@example.com and unattended@example.net to be discarded before any other lookups or redirection.

46.3.12.2 Clone messages to alternate destination (`clonehosts`)

(New in MS 8.0) 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
```

46.3.12.3 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)
```

46.3.12.4 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 `lastresoft`; `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.

46.3.12.5 Multiple gateways on a single channel (`multigate`, `nomultigate`)

The `multigate` channel option tells the MTA to perform an additional rewrite of the envelope To: address during message dequeue processing in order to determine the host to connect to for message delivery. 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 whatever host is specified using the `daemon` channel keyword, and to the *not* the host specified in the message's To: address if `daemon` is not set.

`nomultigate` is the default.

The recipient address is rewriten as if it were a forward-pointing header address, so `$B$F` is typically used in rewrite rule templates intended to only apply during this specific rewrite operation.

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.

46.3.12.6 `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.

46.3.13 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.

46.3.13.1 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 lmtp* channel options; noaddreturnpath is the default for all other channels.

46.3.13.2 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 or new in MS 8.0 the value of whatever attribute is named by the ldap_auth_attr_sender MTA option, 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 46.9 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:

		src-chan|*env-from*|*[resent-]sender*|*[resent-]from*|*auth-originator*
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; that is, when this bit is set then probes take the form *env-from*|*[resent-]sender*|*[resent-]from*|*auth-originator*|*auth-param*
9	512	(New in MS 7.0.5) If set, the final tag set via $T in the *_ACCESS mappings will be prefixed to the AUTH_REWRITE mapping probe; that is, when this bit is set then probes take the form: *ACCESS-tag*|*env-from*|*[resent-]sender*|*[resent-]from*|*auth-originator*

46.3.13.3 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.

46.3.13.4 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 incompliant mail systems that require two digit dates; it should never be used for any other purpose.

46.3.13.5 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 accomodate 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.

46.3.13.6 Host name to use when correcting incomplete addresses (`authhost`, `noauthhost`,`defaulthost`, `nodefaulthost`, `remotehost`, `noremotehost`)

The MTA often receives addresses consisting of a bare local-part and no "@domain" from misconfigured or incompliant mailers and SMTP clients. (Note that the standards do in fact require acceptance of one special address - "postmaster" in SMTP, but do not state how such an address is to be represented in a message header.) This happens often enough that simply disallowing such addresses is rarely an acceptable strategy.

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", producing "mrochek@acme.com").

The set of options described here control how this domain name is selected. In this process the MTA makes a distinction between envelope To (RCPT TO) addresses versus addresses appearing in all other contexts (header, MAIL FROM), as well as distinguishing SUBMIT from SMTP.

In the following table showing what domain is used when various options or protocols are used, *o.org* is the domain attached to the local channel, *a.com* is the domain associated with the current authenticated user's primary email address, *l.com* is the first argument of the `defaulthost` option setting on the current source channel, *r.com* is the second argument

of the `defaulthost` option setting on the current source channel, and *r.edu* is the domain associated with the remote SUBMIT/SMTP client.

Table 46.10 Options controlling missing domain fixups

Option	Protocol	Header/ MAIL FROM	RCPT TO
`authhost` (and authenticated)	SUBMIT/SMTP	@a.com	@a.com
`defaulthost`	SUBMIT/SMTP	@o.org	@o.org
`defaulthost` *l.com*	SUBMIT	@l.com	@l.com
`defaulthost` *l.com*	SMTP	@l.com	@o.org
`defaulthost` *l.com r.com*	SUBMIT	@l.com	@l.com
`defaulthost` *l.com r.com*	SMTP	@l.com	@r.com
`nodefaulthost` (MS 8.0.2.1 or later)	SUBMIT/SMTP	<no fixup>	<no fixup>
`remotehost`	SUBMIT	@r.edu	@r.edu
`remotethost`	SMTP	@r.edu	@o.org
<no options>	SUBMIT/SMTP	@o.org	@o.org

Note: The option in the preceding table are shown in precedence order, that is, a given option setting will override option settings further down the table.

The usual configuration is to have both arguments to `defaulthost` set to the `defaultdomain` on the defaults channel, overriding any use of the local channel host. It may be appropriate in multitenant configurations to set `authhost` on all channels that are marked with the `submit` channel option

Use of the `remotehost` channel option may be considered in rare cases where remote SMTP client exist in another administrative domain which partial addresses to refer to their own users.

Note that rewrite rules can make use of whatever default has been selected via the the `$G` and `$`*`n`*`G` substitutions.

Note that the `switchchannel`, `saslswitchchannel`, `tlsswitchchannel` and various other options can be used to associated incoming SMTP connections with a particular channel, and thus control what set of options are used.

46.3.13.7 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.

46.3.13.8 Envelope tunneling via header fields (`envelopetunnel`))

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 (Tunneling of SMTP Message Transfer Priorities).

46.3.13.9 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.

46.3.13.10 Syntax Error Fixup (`fixsyntaxerrors`, `passsyntaxerrors`)

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 `passsyntaxerrors` 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.

46.3.13.11 Received: from clause content (`forcedreceivedfrom`))

(New in 7.0.5.37) The `forcedreceivedfrom` source channel option is used to specify the text string that appears as the *`source`* within the "`from` *`source`*" clause value in any Received: header fields that are generated. The default, if `forcedreceivedfrom` is not specified, is for the "`from`" clause value to be generated in the usual way from the value of the remote client's HELO/EHLO command and its IP address.

Specifying an empty string as the `forcedreceivedfrom` value causes the "`from`" clause to be suppressed. Note that the "`from`" clause is required by the Received: field syntax specified in RFC 5321 but is optional in the syntax specified in RFC 822.

46.3.13.12 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.

46.3.13.13 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.

46.3.13.14 Controlling Sender Rewriting Scheme (SRS) rewriting in header lines (`headerdecodesrs`, `noheaderdecodesrs`)

New in 8.0.1.3. The `headerdecodesrs` channel option, if set on the current source or destination channel, causes any SRS-encoded addresses found in any address header fields to be decoded. The decoding in this case does not enforce password or timeout checks; it is sufficient that the SRS domain match for decoding to occur.

The default is `noheaderdecodesrs`.

46.3.13.15 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.

46.3.13.16 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 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 MS 6.2 and earlier; as of 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 occassional 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 MS 6.3, the MTA supports the Sieve`editheader` extension, which offers quite a different way to modify message header lines.

46.3.13.17 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.

46.3.13.18 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.

46.3.13.19 Received: from clause content (includereceivedip, suppressreceivedip))

New in 8.0.1.2. The includereceivedip and suppressreceivedip source channel options control the inclusion of IP address information in generated Received: from clauses. The default is includereceivedip.

46.3.13.20 Inner header rewriting (inner, noinner, sourceinner, nosourceinner)

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 and/or the nosourceinner and (new in 8.1.0.1) sourceinner channel options on source channels. noinner and/or nosourceinner 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, or sourceinner, when placed on a source channel, tells the MTA to parse messages and rewrite inner headers.

These options can be applied to any channel.

46.3.13.21 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 langage 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.

46.3.13.22 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 CRLFCRLF 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 CRLFCRLF 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 CRLFCRLF 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.)

46.3.13.23 Automatic splitting of long header lines (`maxheaderaddrs`, `maxheaderchars`)

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 `maxheaderaddrs` channel option controls how many addresses can appear on a single line. The `maxheaderchars` 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.

46.3.13.24 Handling messages that lack any recipient headers (`missingrecipientpolicy`)

RFC 822, the original Internet message format standard, had a requirement that all messages contain at least one recipient header field: a To:, Cc:, or Bcc:.

As of RFC 2822, the original update to RFC 822, relaxed the RFC 822 requirement and allowed submitted messages to be lacking in any recipient header line. This change was carried forward to the current message format standard, RFC 5322.

However, there are still MTAs around that operate according to RFC 822, and in particular may try to be helpful by adding a To: field containing all of the envelope recipients when no recipient fields are present. As such, it may be 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 provides various capabilities that may be useful in addressing this issue. It takes an integer value specifying what approach to use to "fix" messages with no recipient field; 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 MS 6.2 is equivalent to a value of 1 meaning that messages are passed through unchanged---in MS 6.0 and 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 46.11 `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 5322) 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.e.*, ;) To: header line. The phrase used in the group construct is controlled by the `missing_recipient_group_text` 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 `acceptalladdresses` channel option, if used, modifies the timing and form of the rejection.)

Note that the `missing_recipient_policy` MTA option can be used to set an MTA system default for this behavior.

46.3.13.25 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.

46.3.13.26 XBCC SMTP Extension Support (`bccserver`, `nobccserver`)

The `bccserver` channel option, when placed on a SUBMIT server channel, enables the XBCC extension. This extension adds a single XBCC argument to the RCPT TO command. When

present, it marks the corresponding recipient as a blind carbon recipient and the MTA will generate a separate message copy to this recipient and add a Bcc: header field to the copy.

The XBCC argument value may optionally be used to specify a phrase which will preceed the recipient address in the Bcc: header field.

The `nobccserver` channel option disables this extension and is the default.

46.3.13.27 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.

46.3.13.28 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA's SMTP server 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, PORT, DESTADDR, and DESTPORT 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|this-mta's-ip|25|1.2.3.4|12345
```

where note that the values to use in the "source IP" and "source port" fields have been specified via ADDR and PORT, respectively.

Note: Support for DESTADDR and DESTPORT was added in MS 8.0.2.3.

46.3.13.29 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.

46.3.13.30 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 arugment 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.

46.3.13.31 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 46.12 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.

46.3.13.32 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 46.13 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 l channel compared with PMDF version 4.3 and earlier.

46.3.14 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 authhost, noauthhost, defaulthost, nodefaulthost, remotehost, and noremotehost channel options.

46.3.14.1 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 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.

46.3.14.2 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.

46.3.14.3 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...
```

46.3.15 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".

46.3.15.1 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 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).

46.3.15.2 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 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 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.)

46.3.15.3 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 for the various protocols supported by the MTA, including but not limited to SMTP, LMTP, and MTQP.

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.

The ManageSieve server only supports the server subset of the TLS options since there is no ManageSieve client.

`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/MTQP 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/MTQP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP/LMTP/MTQP 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/

MTQP client will attempt TLS use when sending outgoing messages, if sending to an SMTP/LMTP/MTQP server that supports TLS, and `maytlsserver`, which means that the SMTP/LMTP/MTQP 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/LMTP 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. This support is not available in MTQP.

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 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 neglible.)

Note that these options affect only TLS use negotiated at the 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.

46.3.16 ISC channel options

The ISC channel options were removed in MS 8.0.2.2.

See also the ISC options.

46.3.17 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.

46.3.17.1 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.*, *`DATAROOT`*`/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 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. As of MS 8.0.2, a value of 3 causes logging during dequeues without enabling logging for message enqueues. The specific header lines to be logged are controlled by the `log_headers.opt` file, discussed along with the MTA option `log_header`.

When enabling `logheader`, consider also enabling the `log_process` MTA option, as it is helpful for correlating header entries with corresponding regular message entries.

If the goal is to record subsets of information from one or more header lines, rather than necessarily logging full header lines via `logheader`, see as an alternative the (both new-in-MS-8.0) `log_transaction` MTA option and Sieve "`transactionlog`" action.

46.3.17.2 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 enqueuing 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.

Similarly, the TCP/IP channel slave program -- the SMTP server or SMTP SUBMIT server -- will produce multiple files when `slave_debug` is enabled. The first such file produced is a SMTP server or SMTP SUBMIT server log file, with the base name specified via the `logfilename` Dispatcher option for the respective service (by default `tcp_smtp_server.log` and `tcp_submit_server.log`, respectively) with a unique identifying string appended, so typically `tcp_smtp_server.log-`*`uniqueid`* or `tcp_submit_server.log-`*`uniqueid`*. Some debugging regarding general server process initialization is written to such server log files: debugging regarding TLS initialization, the setting of TCP/IP-channel-specific options for the server, *etc*. However, the debugging regarding channel operation handling SMTP connections is written to channel slave log files, with names controlled by the name of the default channel for the service (see the `parameter` Dispatcher option for the service), so have names typically of the form `tcp_local_slave.log-`*`uniqueid`* or `tcp_submit_slave.log-`*`uniqueid`*.

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` messagetrace 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.

46.3.18 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.

46.3.18.1 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 chanmel 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.

46.3.18.2 Maximum allowed recipients or bad commands (recipientlimit, recipientcutoff, deferralrejectlimit, disconnectrecipientlimit, disconnectrejectlimit, disconnectbadcommandlimit, disconnectbadburllimit, disconnectcommandlimit)

The `recipientlimit`, `recipientcutoff` and (new in 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 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 MS 6.2:

```
450 4.7.0 Session bad recipient limit reached; disconnecting
```

or in MS 6.3 and later (MS 6.3 also being when behavior was enhanced so that rejected MAIL FROM's count against the `disconnectrejectlimit`, whereas in MS 6.2 only rejections at the RCPT TO or VRFY stages counted; 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 MS 6.3 and later, potentially after a failed VRFY attempt). (Note that because the disconnect usualy 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 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.

46.3.18.3 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 on a per-source-channel basis 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 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 earler 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.

46.3.18.4 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.

46.3.19 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.

46.3.19.1 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 apppeared 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.

46.3.20 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.

46.3.20.1 Message Tracking and Recall Channel Options (`notrackingclient`, `notrackingserver`, `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 `notrackingserver` 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 `notrackingclient` 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.)

46.3.20.2 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 messag-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 infeasilbility of computing X given T, where SHA1(X) = MD4(T).

46.3.21 MLS channel options

RESTRICTED: Not yet fully implemented. There are a few channel options relating to MLS (Multi Layer Security).

46.3.21.1 MLS (Multi Layer Security) Channel options: `mlslabel` (string), `mlsrange` (string)

RESTRICTED: Not yet fully implemented.

46.3.22 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.

46.3.22.1 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.

46.3.22.2 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`.

46.3.22.3 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`.

46.3.22.4 Notification and disposition channels (`dispositionchannel`, `notificationchannel`)

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.

46.3.22.5 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.

46.3.22.6 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 langage 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.

46.3.22.7 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.)

46.3.22.8 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, l, 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.

46.3.22.9 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.

46.3.22.10 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.

46.3.22.11 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 NOTARY[g] 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.

46.3.22.12 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.

46.3.23 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.

46.3.23.1 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.

The `addrsperjob` 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 `addrsperjob` 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 `addrsperjob 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 `addrsperjob` 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.

46.3.23.2 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. `secondclassfter` 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.

46.3.23.3 Delivery retry intervals (`backoff`, `urgentbackoff`, `normalbackoff`, `nonurgentbackoff`, `ipbackoff`)

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.

Note that all of the letters in the value must be written in upper case.

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.

Additionally, as of MS 8.0.2.3, an additional `ipbackoff` option is provided to set backoff times for messages that have been placed in IP backoff mode. The ability to set ipbackoff values on a per-domin basis is also provided by the ipbackoff smartsend parameter.

The default intervals between delivery retry 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
ip: 60, 120, 120, 240, 240, 240, 480
```

Note that setting the various options on the defaults channel can be used to override these built-in defaults, as opposed to having to set the options on every channel with an associated queue in the configuration, e.g.,

```
msconfig> set channel:defaults.backoff "PT30M PT1H PT2H PT4H"
```

Also note that the MTA has special handling of the case of problems delivering to *some* recipients of multi-recipient messages: in such cases, the failing recipients are eligible for another "immediate" delivery attempt without regard to `*backoff` setting; see the discussion of MTA message transaction log file "Z" records.

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.

46.3.23.4 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`.

46.3.23.5 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 implemention 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.

46.3.23.6 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 on a per-source-channel basis 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 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 earler 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.

46.3.23.7 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 `submit` 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 `deferreddestination` 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 `deferreddestination` option on a destination channel opens the door to anyone submitting a message to that channel that will 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.

46.3.23.8 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.

46.3.23.9 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".

46.3.23.10 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.

46.3.23.11 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). New in Cayenne, see the `envelopetunnel` channel option for "tunneling" MT-PRIORITY, via a header field, through systems that do not support the MT-PRIORITY SMTP extension, as described in RFC 6758 (Tunneling of SMTP 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.

46.3.23.12 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.

Note that for iMS/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.

46.3.23.13 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 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.

46.3.23.14 `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.

46.3.24 Sensitivity limits channel options

Several channel options set channel-specific message sensitivity limits.

46.3.24.1 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. (Also note that the `acceptalladdresses` channel option, if used, postpones the timing of the message rejection.)

The `log_sensitivity` MTA option permits logging of the sensitivity of messages passing through the MTA.

46.3.25 Sieve filters and delivery flags channel options

There are a number of channel options controlling Sieve filters and message envelope and delivery flags.

46.3.25.1 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 informtion, the MTA's delivery flag transfer facilities may be used to transport this information between MTAs; see the `deliveryflags` channel option.

46.3.25.2 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 46.14 `deliveryflags` MTA option bit values

Bit	Value	Usage
0	1	Interpret subaddresses as folder names for delivery. This bit is also set by Sieve "`fileinto`" actions.
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. (The `tcp_tas` channel, for example, has this set.)
2	4	Reserved for internal use only.
3	8	Reserved for internal use only.
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 `discard` or `jettison` action and enqueueing to the `filter_discard` 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 source channel for an SMTP/LMTP server, enable transfer of delivery flags; equivalent to the `flagtransfer` channel option
7	128	Messages enqueued to the channel are considered, for purposes of `CONVERSIONS` mapping table testing, to have a conversion tag set. This bit is set automatically when it is needed and is not intended to be set using the channel option.
8	256	Reserved for internal use only.
9	512	Handle as if SMTP AUTH (SASL) had been used as far as access checks are concerned. This bit is set automatically when it is needed and is not intended to be set using the channel option.
10	1024	Handle as if TLS had been used as far as access checks are concerned.
11	2048	Handle as if address produced by alias as far as access checks are concerned.
12	4096	Bypass all list access checks.

All remaining bits in this option are reserved for internal use and should not be set. 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. As of MS 8.0.2.3, `flagtransfer` also causes the SMTP server (but not the LMTP server) to advertise the XCONVTAG extension.

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) delivery flags, IMAP flags, and (in the case of SMTP) conversion tags.

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.

Some of the delivery flag bits affect server security, as such, the `flagtransfer` channel option should never be set on a channel that is exposed to untrusted traffic.

Setting `flagtransfer` is equivalent to setting bit 6 (value 64) of the `deliveryflags` channel option.

46.3.25.3 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 Sieve 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 Sieve filters for that channel. It takes a required URL argument describing the users' Sieve filter location -- typically a (templated, making use of substitution based on user) 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).

IMPORTANT NOTE: Due to the way the MTA caches Sieves internally - the URL is used as a cache key - it's vital that each user have a distinct Sieve file or be associated with a distinct LDAP entry when the `filter` channel option is used. Failure to insure this separation will cause Sieve ownership checks to behave in unexpected ways and may allow users to bypass certain security checks.

The `sourcefilter` and `destinationfilter` channel options may be used on general MTA channels to specify a channel-level Sieve 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 enqueuing a message to the destination channel on which it is specified.) These channel options take a required URL argument describing the channel Sieve filter location (typically a file). `nosourcefilter` and `nodestinationfilter` are the defaults and mean that no channel Sieve filter is enabled for either direction of the channel.

The obsolete `channelfilter` and `nochannelfilter` channel options are synonyms for `destinationfilter` and `nodestinationfilter`, respectively.

IMPORTANT NOTE: It's very important not to confuse the `filter` and `destinationfilter` channel options. The critical difference is that `destinationfilter` Sieves are system-level whereas `filter` Sieves are user-level. As such, a system-level Sieve invoked with `filter` may not have access to the capabiities it needs, while a user-level Sieve invoked with `destinationfilter` would give the user control over various aspects of message disposition handling they should not be able to change.

New in Messaging Server 7.0 update 3 are the `disablesourcefilter` and `disabledestinationfilter` channel options. These options can be used to suppress the evaluation and interpretation of Sieve filters based on source or destination channel, respectively. Each option takes a single nonnegative integer argument, whose value is interpreted as follows:

Table 46.15 `disablesourcefilter` and `disabledestinationfilter` 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 `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), are evaluated and interpreted

4	Spam filter, source channel, the `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), and destination channel Sieves are evaluated and interpreted
5	Spam filter, source channel, the `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), destination channel, and `ORIG_SEND_ACCESS` Sieves are evaluated and interpreted
6	Spam filter, source channel, the `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), destination channel, `ORIG_SEND_ACCESS`, and `SEND_ACCESS` Sieves are evaluated and interpreted
7	Spam filter, source channel, the `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), destination channel, `ORIG_SEND_ACCESS`, `SEND_ACCESS`, and `ORIG_MAIL_ACCESS` Sieves are evaluated and interpreted
8	Spam filter, source channel, the `systemfilter` MTA system Sieve (which in legacy configuration was `imta.filter`), destination channel, `ORIG_SEND_ACCESS`, `SEND_ACCESS`, `ORIG_MAIL_ACCESS` and `MAIL_ACCESS` Sieves are evaluated and interpreted

46.3.25.4 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_lmtpc*` 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.

46.3.25.5 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.

46.3.26 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.

46.3.26.1 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 chanmel 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.

46.3.26.2 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/mailing list limit imposed via the attribute named by the `ldap_maximum_message_size` MTA option, or per-group/mailing 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 receipient information), but will nevertheless be logged as a "J" record (missing some fields such as recipient fields) as of 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".

(Note also that the `acceptalladdresses` channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

46.3.26.3 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 on a per-source-channel basis 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 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 earler 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.

46.3.26.4 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.

46.3.27 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,...,8` as well.

See also the Spamfilter MTA options for configuration of the location and operation of the spam/virus filter package interfaces. Note that "opt in" to spam/virus filter package processing may also be selected on a per-user, per-domain, or per-alias basis.

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`.

46.3.27.1 `destinationspamfilterN`, `destinationspamfilterNoptin`, `sourcespamfilterN`, `sourcespamfilterNoptin`, `disabledestinationspamfilterN`, `disablesourcespamfilterN` Channel Options

The `destinationspamfilterN`, `destinatationspamfilterNoptin`, `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 "opt-in" value or values set, as some spam/virus filter packages (such as Brightmail) support different "choices" or "opt-in" values for what type of filtering will be performed (*e.g.*, `spam` vs. `virus`). The MTA supports use of multiple, comma-separated "opt-in" values as an argument to an `*optin` channel option. Spam/virus filter package processing may also be triggered, with a specific "opt-in" value or values, via use of a per-user attribute (see the `ldap_optinN` and `ldap_source_optinN` MTA options) or a per-domain attribute (see the `ldap_domain_attr_optinN` MTA options). When "opt-in" values from multiple sources apply for a message -- for instance, if an "opt-in" value is set via a

channel option as well as via an attribute---the MTA will pass all the applicable "opt-in" values to the spam/virus filter package.

The `sourcespamfilter*` channel options enable spam/virus filter package processing based upon source channel; when such a channel option applies, the spam/virus filter package will be activated towards the start of the SMTP dialogue (MAIL FROM stage). (For even earlier spam/virus filter package activation, see Spamfilter early verdicts.) The `destinationspamfilter*` channel options enable spam/virus filter package processing based upon destination channel, which is determined for each recipient at the RCPT TO stage of the SMTP dialogue; so if a `destinationspamfilter*` channel option applies, that spam/virus filter package activation occurs at RCPT TO stage.

The `disable*spamfilter`*N* channel options can be used to disable, on a channel-specific basic, spam/virus filtering that would otherwise be performed. In particular, note that the `disable*spamfilter`*N* channel options override any user-level opt-in to spam/filter package processing (as for instance selected via an LDAP attribute named by an `ldap_optin`*N* MTA option), and override any channel level opt-in (as for instance via a `*spamfilter`*N*`optin` channel option). For instance, one might use `disablesourcespamfilter1` 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).

As of MS 8.0.1.3, the $+^ flag in the `FROM_ACCESS` mapping table can also be used to disable source channel spam filter optins. Also as of 8.0.1.3, the same $+^ flag in a recipient access mapping table can be used to disable any active spam filter optins at the current aliasing level. In either case the flag takes a comma separated list of spam filters to disable.

46.3.28 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 TCP/IP connections and DNS lookups channel options.

46.3.28.1 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.

46.3.28.1.1 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 `$D`*`channel-name`* or `$d`*`channel-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).

46.3.28.2 XBCC SMTP Extension Support (`bccserver`, `nobccserver`)

The `bccserver` channel option, when placed on a SUBMIT server channel, enables the XBCC extension. This extension adds a single XBCC argument to the RCPT TO command. When present, it marks the corresponding recipient as a blind carbon recipient and the MTA will generate a separate message copy to this recipient and add a Bcc: header field to the copy.

The XBCC argument value may optionally be used to specify a phrase which will preceed the recipient address in the Bcc: header field.

The `nobccserver` channel option disables this extension and is the default.

46.3.28.3 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.

46.3.28.4 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 incompliant 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-incompliant 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 incompliant intermediary.

46.3.28.5 Recipient validity date check (checkrrvs, ignorerrvs)

The checkrrvs and ignorerrvs source channel options, for support of RFC 7293, are new in MS 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.

46.3.28.6 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 tranfer 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.

46.3.28.7 Channel operation type (submit, relay, passthrough, conditionalpassthrough, conditionalrelay, destinationpassthrough, destinationrelay)

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 `mustsaslserver` and `maytlsserver` (or `musttlsserver` 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 can be a 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 relay. 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.

New in MS 8.0, the `destinationpassthrough` channel option, if set on a destination channel, causes any enqueue of a message to that channel to be done in passthrough mode

as long as the source channel isn't itself marked with the submit channel keyword. If submit mode is engaged the destinationpassthrough channel option is a no-op.

The default is `destinationrelay`, meaning the destination channel does not activate passthrough 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-type`, 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.

A common use-case for passthrough mode is to prevent message modification after DKIM signing. This is normally done by introducing an additional channel hop so that normal message rewriting, canonicalization, and so on can take place, attaching an appropriately configured DKIM signing milter to that channel, and marking the channel with the `passthrough` channel option to prevent any additional modifications. For example, assuming that all traffic from the tcp_submit channel to the tcp_local should be signed, an appropriate DKIM signing milter is set up on spam filter slot 3, the following configuration could be used:

Conversion mapping:

```
CONVERSION

  IN-CHAN=TCP_SUBMIT;OUT-CHAN=TCP_LOCAL;CONVERT  YES,CHANNEL=PROCESS-DKIM-SIGN
```

Channel definition:

```
process-dkim-sign sourcespamfilter3 passthrough \
 noreceivedfor noreceivedfrom enqueueremoveroute
dkim-sign.example.com
```

46.3.28.8 Maximum allowed recipients or bad commands (`recipientlimit`, `recipientcutoff`, `deferralrejectlimit`, `disconnectrecipientlimit`, `disconnectrejectlimit`, `disconnectbadcommandlimit`, `disconnectbadburllimit`, `disconnectcommandlimit`)

The `recipientlimit`, `recipientcutoff` and (new in 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 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 MS 6.2:

```
450 4.7.0 Session bad recipient limit reached; disconnecting
```

or in MS 6.3 and later (MS 6.3 also being when behavior was enhanced so that rejected MAIL FROM's count against the `disconnectrejectlimit`, whereas in MS 6.2 only rejections at the RCPT TO or VRFY stages counted; 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 MS 6.3 and later, potentially after a failed VRFY attempt). (Note that because the disconnect usualy 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 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 maxprotocolerrorsIMAP or POP option.

46.3.28.9 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 46.16 deliveryflags MTA option bit values

Bit	Value	Usage
0	1	Interpret subaddresses as folder names for delivery. This bit is also set by Sieve "fileinto" actions.
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. (The tcp_tas channel, for example, has this set.)
2	4	Reserved for internal use only.
3	8	Reserved for internal use only.
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 discard or jettison action and enqueueing to the filter_discard 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 source channel for an SMTP/LMTP server, enable transfer of delivery flags; equivalent to the flagtransfer channel option
7	128	Messages enqueued to the channel are considered, for purposes of CONVERSIONS mapping table testing, to have a conversion tag set. This bit is set automatically when it is needed and is not intended to be set using the channel option.
8	256	Reserved for internal use only.
9	512	Handle as if SMTP AUTH (SASL) had been used as far as access checks are concerned. This bit is set automatically when it is needed and is not intended to be set using the channel option.

10	1024	Handle as if TLS had been used as far as access checks are concerned.
11	2048	Handle as if address produced by alias as far as access checks are concerned.
12	4096	Bypass all list access checks.

All remaining bits in this option are reserved for internal use and should not be set. 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. As of MS 8.0.2.3, `flagtransfer` also causes the SMTP server (but not the LMTP server) to advertise the XCONVTAG extension.

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) delivery flags, IMAP flags, and (in the case of SMTP) conversion tags.

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.

Some of the delivery flag bits affect server security, as such, the `flagtransfer` channel option should never be set on a channel that is exposed to untrusted traffic.

Setting `flagtransfer` is equivalent to setting bit 6 (value 64) of the `deliveryflags` channel option.

46.3.28.10 Limiting time to deliver (`deliverbychannel`))

The `deliverbychannel` source channel option specifies the source channel used for the generation of nondelivery receipts associated with the DELIVERYBY SMTP extension in the absence of any other information. The default is to use the l channel if this option is not specified. Note that there should be few if any cases where this option should be set.

46.3.28.11 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.

46.3.28.12 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.

46.3.28.13 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 MS 6.3) `error_text_transaction_limit_exceeded` MTA option.

`disconnecttransactionlimit`: new in 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.

46.3.28.14 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.

46.3.28.15 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. (Note that the timing of the rejection can be postponed via the `acceptalladdresses` channel option.)

The MS 8.0.2 release adds MTA support for EAI messages and the SMTPUTF8 extension. EAI messaging is documented in RFC 6530 (overview), RFC 6531 (SMTPUTF8 extension), RFC 6532 (header format changes), and RFC 6533 (DSN and MDN format changes). Three additional channel options have been added to enable and control EAI support:

`utf8header` As a SMTP source channel option, offer the SMTPUTF8 SMTP extension. As a destination channel option, allow enqueue and dequeue of EAI messages unconditionally; in particular, the SMTPUTF8 extension will not be required. Note that delivery of EAI messages via SMTP/LMTP to a non-EAI system is a standards violation.

`utf8negotiate` As a SMTP source channel option, offer the SMTPUTF8 SMTP extension. As a destination channel option, allow enqueue unconditionally. On dequeue, require the SMTPUTF8 extension be offered by SMTP/LMTP servers for EAI messages with EAI recipient (RCPT TO) addresses or un-downgradable EAI originator (MAIL FROM) addresses.

`utf8strict` Same as `utf8negotiate` as far as SMTP/LMTP destination channels are concerned. But additionally, for SMTP/LMTP servers (source channels), reject messages that contain 8bit headers without having negotiated the SMTPUTF8 extension and 8bit bodies without having negotiated the 8BITMIME extension. (Note that such rejections are postponed if the `acceptalladdress` channel option is used.)

As of MS 8.0.2.2 the channel default for internal channels has changed from `eightnegotiate` to `utf8always`. Note, however, that the channel default remains `eightnegotiate` on all other channels; EAI support must therefore be explicitly enabled. A channel set to `eightnegotiate` will not offer the SMTPUTF8 extensions or allow EAI messages with EAI recipient or originator addresses to be enqueued. Also note that the Message Store does not support EAI at the present time.

46.3.28.16 Responding to SMTP EXPN commands (`expnallow`, `expndefault`, `expndisable`)

New in 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` TCPIP-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.

46.3.28.17 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 `submit` 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

deferreddestination 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 deferreddestination option on a destination channel opens the door to anyone submitting a message to that channel that will 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.

46.3.28.18 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---should be set explicitly on all SMTP channels; but if no such option is set on a tcp_* channel, the channel will default to

smtp_crorlf. 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 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, saslswitchchannel, tlsswitchchannel, 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.

46.3.28.19 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).

46.3.28.20 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-existant 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 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 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.

46.3.28.21 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 accomodate 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.

46.3.28.22 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). New in Cayenne, see the `envelopetunnel` channel option for "tunneling" MT-PRIORITY, via a header field, through systems that do not support the MT-PRIORITY SMTP extension, as described in RFC 6758 (Tunneling of SMTP 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.

46.3.28.23 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.)

46.3.28.24 Proxy protocol support

New in MS 8.1.0.1, the `proxyprotocol` enables server-side use of the proxy protocol. Note that the proxy protocol is not negotiated; enabling it on the server side will cause connections to hang if the client doesn't send the necessary PROXY command. Presently, only proxy protocol v1 (text format) is supported.

At present only the SMTP and MMP servers support the proxy protocol for incoming connections. This is supported with SMTP and Submission protocols (port 25 & 587), but works incorrectly with the submissions protocol (port 465) so it's important to make sure any slave channel used to offer submissions service is configured with noproxyprotocol.

46.3.28.25 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.

46.3.28.26 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).

46.3.28.27 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA's SMTP server 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, PORT, DESTADDR, and DESTPORT 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|this-mta's-ip|25|1.2.3.4|12345
```

where note that the values to use in the "source IP" and "source port" fields have been specified via ADDR and PORT, respectively.

Note: Support for DESTADDR and DESTPORT was added in MS 8.0.2.3.

46.3.28.28 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 MS 6.0 inserts a header line

```
Sun-ONE-SMTP-Warning: Lines longer than SMTP allows found and truncated.
```

or in 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 normally causes the MTA to reject such illegal messages with a

```
550 5.6.0 lines longer than SMTP allows encountered; message rejected
```

SMTP error (but see the `acceptalladdresses` channel option which postpones the error generation), 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 MS 6.0)

```
Sun-One-SMTP-Warning: Lines longer than SMTP allows found and wrapped.
```

or in 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 `saslswitchchannnel` channel option effect, or a `mailSMTPSubmitChannel` LDAP attribute effect).

46.3.28.29 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 announcment.

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.

46.3.28.30 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.

46.3.29 TCP/IP connections and DNS lookups channel options

A number of channel options affect TCP/IP connections and DNS lookups.

46.3.29.1 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.

46.3.29.2 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.

46.3.29.3 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)
```

46.3.29.4 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 MS 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.)

Whether the operating system's TCP/IP local host tables are used in addition to the DNS for lookups is generally controlled by `/etc/nsswitch.conf` or the equivalent. Generally, operating system distributions are configured so that local host tables will indeed be consulted.

When nameserver lookups are being performed, that is, unless the `nsswitch.conf` file 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). Futhermore, 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.

46.3.29.5 Name lookup failure handling(`dnsforcetemporary`, `nodnsforcetemporary`)

When attempting to make an outgoing connection, name lookups that return a "host not found" or "no addresses" sorts of errors are normally treated as permanent failures. The `dnsforcetemporary` channel option, which places on the dequeuing channel, alters this behavior and causes this errors to be treated as temporary. `nodnsforcetemporary` is the default.

Important note: This restricted option is only intended for use in special situations where the list of possible remote hosts is tightly controlled. Use in other situatiions, especially ones where users are able to specify the remote host, will lead to delays in reporting failures conditions.

46.3.29.6 Reverse DNS and IDENT lookups on incoming SMTP connections (`identtcp`, `identtcplimited`, `identtcpnumeric`, `identtcpsymbolic`, `identnone`, `identnonelimited`, `identnonenumeric`, `identnonesymbolic`, `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 `identnonesymbolic` 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 46.17 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
`identtcp`	Yes	Yes	Yes	Yes	Yes

identtcplimited	Yes	Yes	Yes	Yes	No
identtcpnumeric	Yes	No	Yes	No	No
identtcpsymbolic	Yes	Yes	No	Yes	Yes
identnone	No	Yes	Yes	Yes	Yes
identnonelimited	No	Yes	Yes	Yes	No
identnonenumeric	No	No	Yes	No	No
identnonesymbolic	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.

46.3.29.7 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.

As of MS 8.0.2.3 two different addresses, one used for logging and the other as the actual TCP/IP source address, can be specified. Normally the same address is used for both purposes, but in cases where NAT or other forms of address translation are in use it may be useful to have the actual IP address that appears to external agents appear as the address in transport information fields in the logs. The address that's actually used does need to be logged somewhere, so it is appended to the application information field with the usual "/" separator, prefixed with a sharp sign.

When two addresses are specified they must be separated by a sharp sign with the logging address appearing first, i.e., "logging-address#bind-address".

46.3.29.8 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 `lastresoft`; `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.

46.3.29.9 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-existant 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 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 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.

46.3.29.10 Maximum rate to connect to a domain (`maxconnectionrateperdomain`)

The `maxconnectionrateperdomain` smartsend channel option provides the means to limit the rate at which connections are made from the channel. The value of this option is one or two space-separated rate limit lists. The first limit list value specifies the rate limits to use when things are operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

Note that the identically named `maxmessagerateperdomain` smartsend parameter provides the same functionality at the domain or IP level.

This option requires that the smartsend callouts, in particular the callout from the AUTH_ACCESS mapping, be installed. This option will be ignored if smartsend is not configured.

This option also currently requires that a memcache or Redis protocol server be configured at either the MTA or channel level.

46.3.29.11 Maximum number of connections from an IP to a domain (maxconnectionsperdomain)

The maxconnectionsperdomain smartsend channel option implements limits on the number of simultaneous connections that can be opened to a destination domain. The value of this parameter is one or two space-separated limit lists. The first value specifies the limits to use when the IP address is operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

Note that the identically named maxconnectionsperdomain smartsend parameter provides the same functionality at the domain or IP level.

This option requires that the smartsend callouts, in particular the callouts from the AUTH_ACCESS and AUTH_DEACCESS mappings, be installed. This option will be ignored if smartsend is not configured.

46.3.29.12 Maximum rate to send to a domain (maxmessagerateperdomain)

The maxmessagerateperdomain smartsend channel option provides the means to limit the rate at which messages are delivered from the channel queue. The value of this option is one or two space-separated rate limit lists. The first limit list value specifies the rate limits to use when things are operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

Note that the identically named maxmessagerateperdomain smartsend parameter provides the same functionality at the domain or IP level.

This option requires that the smartsend callouts, in particular the callout from the AUTH_ACCESS mapping, be installed. This option will be ignored if smartsend is not configured.

This option also currently requires that a memcache or Redis protocol server be configured at either the MTA or channel level.

46.3.29.13 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).

46.3.29.14 `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.

46.3.29.15 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 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 (`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.

46.3.29.16 SPF DNS lookups (spfhelo, spfmailfrom, spfnone, spfrcptto)

New in MS 6.3-0.15. The spfhelo, 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 spfhelo 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 spfhelo is set):

```
451 4.4.3 Temporary error in SPF verification of HELO domain
```

or a permanent error such as (in this example, when spfhelo 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 spfhelo 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 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.

46.3.29.16.1 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
```

46.3.29.17 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 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.

46.3.30 TLS and SASL channel options

A number of channel options relate to SASL (SMTP AUTH) and/or TLS use.

For global MTA configuration, see also Password and TLS MTA options. And for general Messaging Server configuration of authentication and certificate handling, see the Auth options, various base.auth*, base.ssl*, and base.tls* options (and for LDAP connections, the ugldapusessl and ldaprequiretls base options), the Base certmap options, and the sectoken options.

46.3.30.1 authusername Option Under channel

The authusername channel option is described in the externalidentity section.

46.3.30.2 `authpassword` Option Under channel

The `authpassword` channel option is described in the `externalidentity` section.

46.3.30.3 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.)

See the Base certmap options for general configuration of certificate mapping, as needed for EXTERNAL authentication via client certificates.

46.3.30.4 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 or new in MS 8.0 the value of whatever attribute is named by the `ldap_auth_attr_sender` MTA option, 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 46.18 `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:

		src-chan|env-from|[resent-]sender|[resent-]from|auth-originator
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; that is, when this bit is set then probes take the form *env-from|[resent-]sender|[resent-]from|auth-originator|auth-param*
9	512	(New in MS 7.0.5) If set, the final tag set via $T in the *_ACCESS mappings will be prefixed to the AUTH_REWRITE mapping probe; that is, when this bit is set then probes take the form: *ACCESS-tag|env-from|[resent-]sender|[resent-]from|auth-originator*

46.3.30.4.1 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:

env-from|[resent-]sender|[resent-]from|auth-originator

If (new in MS 6.2) bit 5 (value 32) is set in the the authrewrite argument, the probe is prefixed with the source channel and a vertical bar; if (new in MS 7.3-11.01) bit 8 (value 256) is set in the authrewrite argument, the probe is suffixed with a vertical bar and the AUTH parameter from the MAIL FROM command; if (new in MS 7.0.5) bit 9 (value 512) is set in the authrewrite argument, the probe is prefixed with the final tag set by the FROM_ACCESS or recipient address *_ACCESS mapping tables and a vertical bar. Thus with these three optional bits set, the probe has the format:

ACCESS-tag|src-chan|env-from|[resent-]sender|[resent-]from|auth-originator|auth-param

New in MS 8.0.2.3, the authrewrite_extra_headers MTA option may be used to specify additional header fields to include in the mapping probe. These fields always appear at the end of the probe, separated by vertical bars.

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 access checking is done using the AUTH_REWRITE mapping in order to prevent envelope From forgery by authenticated users.

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 MS 7.0, if a $A is specified, then its argument is interpreted as a header line to add to the primary message header.

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 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 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. (Note that use of the `acceptalladdresses` channel option postpones the rejection.)

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 paragraphs above; for instance,

JYZA$O*env-from*|*sender*|*from-hdr*|*hdr-line-to-add*|*auth-param*

or

XN|*error-code*|*rejection-text*

Technically, one could use all seven flags in the same entry, though it does not seem likely to be useful (as in particular the changes to the message, such as changes of address or addition of a header line) do not apply since the rejection is going to occur at the SMTP protocol level):

JYZAOX$N*env-from*|*sender*|*from-hdr*|*hdr-line-to-add*|*auth-param*|*error-code*|*rejection-text*

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

AUTH_REWRITE mapping flags summarizes the available (output) flags and input flags.

Table 46.19 AUTH_REWRITE mapping flags

Flag	Description
$*	(New in MS 7.0u2) Force disconnect of the SMTP session
Flags with arguments, in argument reading order[1]	
$J*address*	Override original envelope From with specified *address*
$K*address*	Override original envelope From with specified *address*
$T	Force addition of a Sender: *address* or Resent-Sender: *address* header line
$Y	Force addition of a Sender: *address* or Resent-Sender: *address* header line
$Z*address*	Force replacement of the original From: (or Resent-From:) header line with From: *address* or Resent-From: *address*; note this is **NOT RECOMMENDED**, as it is **CONTRARY TO INTERNET STANDARDS** and quite likely to **HARM** the overall security of your users. It should almost **NEVER** be done--- *THIS MEANS YOU!*
$A*header-line*	Add the specified *header-line*
$O*AUTH-parameter*	(New in MS 7.0u3) Use the specified *AUTH-parameter* as the new value to relay as the AUTH value for this message in MAIL FROM.
$<*syslog-text*	(New in MS 8.0.2.3) Send the specified text to syslog.
$>*syslog-text*	(New in MS 8.0.2.3) Send the specified text to syslog if $N is also specified.
$>*syslog-text*	(New in MS 8.0.2.3) Send the specified text to syslog if $N is also specified.
$D*dkim-parameters*	(New in MS 8.1.0.1) Specify one or more comma-separated DKIM signing parameters to enable DKIM signing. The format is the same as that used in the CONVERSIONS mapping.
$H*received-domain*	(New in MS 8.1.0.1) Specify a value to override the current received_domain option setting.
$I*id-domain*	(New in MS 8.1.0.1) Specify a value to override the current id_domain option setting.
$+L*header1,header2,...*	(New in MS 8.1.0.1) Specifies a comma-separated list of header fields to include in transaction log records that are generated.
$X*x.y.z*	(New in MS 6.3) Set the extended error code to *x.y.z*, in place of the default of 5.7.0. $X is only active if $N is also specified.

`$N`*`error-text`*	(New in MS 6.2) Reject the message, optionally specifying *`error-text`* to use in place of the default text "`invalid original address used`".
Input flag comparisons	Description
`$:|`	(New in an MS 7.0 patch) Match only if external material (*e.g.*, an envelope address) in the probe contained a vertical bar
`$;|`	(New in an MS 7.0 patch) Match only if no vertical bars were present in any external material in the probe
`$:A`	(New in MS 8.0) Match only if SASL authentication (SMTP AUTH) has succeeded
`$;A`	(New in MS 8.0) Match only if SASL authentication (SMTP AUTH) has not been used, or if attempted has not succeeded
`$:E`	(New in MS 8.0) Match only if EHLO (ESMTP) was used
`$;E`	(New in MS 8.0) Match only if EHLO (ESMTP) was not used
`$:L`	(New in MS 8.0) Match only if LHLO (LMTP) was used
`$;L`	(New in MS 8.0) Match only if LHLO (LMTP) was not used
`$:P`	(New in MS 8.0) Match only if POP-before-SMTP was used
`$;P`	(New in MS 8.0) Match only if POP-before-SMTP was not used
`$:R`	(New in MS 8.0) Match if the current, enqueueing channel is an "internal" channel such as the reprocess channel
`$;R`	(New in MS 8.0) Match if the current, enqueueing channel is something other than an "internal" channel
`$:T`	(New in MS 8.0) Match only if a SSL/TLS security layer has been negotiated
`$;T`	(New in MS 8.0) Match only if SSL/TLS was not used

[1] To use multiple flags with arguments, separate the arguments with the vertical bar character, `|`, placing the arguments in the order listed in this table.

An example of an AUtH_REWRITE mapping that prevents authenticated users from sending messages from addresses other than the ones listed in their mail, mailAltnerateAddress, or mailEquivalentAddress mappings would be:

```
AUTH_REWRITE

! Probe format: env-from|[resent-]sender|[resent-]from|auth-originator
! Check to see if the From: matches the authenticated user's mail attribute;
! exit with success if it does
     *|*|*|$2*   $E
! Fetch the base DN we'll need to look up the user
     *|*@*|*@*   $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4
! Probe now: BASE|base-DN|from|auth-originator
! Check to see if the From: matches the authenticated user's
! mailAlternateAddress or mailEquivalentAddress attributes.
     BASE|*|*|*  $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(|(mailAlternateAddress=$=$1$_)(mailEquivalentAddress=$=$1$_)))
! Exit with success if it does
     FOUND|*|*|* $E
! Fail any other From: address
     *           $NFrom$ address$ is$ not$ one$ of$ your$ addresses.
```

Note that the `authrewrite` source channel option needs to be set to 3 for the AUTH_REWRITE mapping to be invoked.

A more sophisticated version of this mapping that allows users to specify arbitrary subaddresses would be:

```
REMOVE_SUBADDRESS

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$0@$2
  "$_*+$_*"@*                           $Y"$0"@$2
  $_*+$_*@*                             $Y$0@$2
  *                                     $Y$0

REMOVE_SUB_LDAP_QUOTE

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$=$0@$2$_
  "$_*+$_*"@*                           $Y$="$0"@$2$_
  $_*+$_*@*                             $Y$=$0@$2$_
  *                                     $Y$=$0$_

MATCH_NOSUBADDRESS

  *|*   $C$|REMOVE_SUBADDRESS;$0||$|REMOVE_SUBADDRESS;$1|
  *|$0* $Y

AUTH_REWRITE

! Check to see if the From: matches the authenticated user's mail attribute;
! exit with success if it does
  *|*|*|* $C$|MATCH_NOSUBADDRESS;$2$|$3|$E
! Fetch the base DN we'll need to look up the user
  *|*@*|*@* $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4
! Check to see if the From: matches the authenticated user's
! mailAlternateAddress or mailEquivalentAddress attributes.
  BASE|*|*|* \
  $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(|(mailAlternateAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(mailEquivalentAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)))[|$1|$2
! Exit with success if it does
  FOUND|*|*|* $E
! Fail any other From: address
  * $NFrom$ address$ is$ not$ one$ of$ your$ addresses.
```

An even more sophisticated approach that provides "send on behalf of" functionality is possible. Assuming a sendOnBehalfOf attribute has been added to the user's entry listing the addresses the user is authorized to send on behalf of, the following mapping could be used:

```
REMOVE_SUBADDRESS

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$0@$2
  "$_*+$_*"@*                           $Y"$0"@$2
  $_*+$_*@*                             $Y$0@$2
  *                                     $Y$0

REMOVE_SUB_LDAP_QUOTE

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$=$0@$2$_
  "$_*+$_*"@*                           $Y$="$0"@$2$_
  $_*+$_*@*                             $Y$=$0@$2$_
  *                                     $Y$=$0$_

MATCH_NOSUBADDRESS

  *|*   $C$|REMOVE_SUBADDRESS;$0||$|REMOVE_SUBADDRESS;$1|
  *|$0* $Y

AUTH_REWRITE

! Check to see if the From: matches the authenticated user's mail attribute;
! exit with success if it does
  *|*|*|* $C$|MATCH_NOSUBADDRESS;$2$|$3|$E
! Fetch the base DN we'll need to look up the user
  *|*@*|*@* $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4
! Check to see if the From: matches the authenticated user's
! mailAlternateAddress or mailEquivalentAddress attributes.
  BASE|*|*|* \
  $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(|(mailAlternateAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(mailEquivalentAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(sendOnBehalfOf=$1)))[|$1|$2
! Exit with success if it does
  FOUND|*|*|* $E
! Fail any other From: address
  * $NYou$ are$ not$ authorized$ to$ send$ from$ the$ address$ you$ specified
```

If the "send on behalf of" permissions are instead stored on the "other side" - in, say, an, mailGrantSendPermissionsTo on the granting user's entry that specifies the mail attribute of the user permissions are being granted to, the following mapping could be used:

```
REMOVE_SUBADDRESS

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$0@$2
  "$_*+$_*"@*                           $Y"$0"@$2
  $_*+$_*@*                             $Y$0@$2
  *                                     $Y$0

REMOVE_SUB_LDAP_QUOTE

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$=$0@$2$_
  "$_*+$_*"@*                           $Y$="$0"@$2$_
  $_*+$_*@*                             $Y$=$0@$2$_
  *                                     $Y$=$0$_

MATCH_NOSUBADDRESS

  *|*   $C$|REMOVE_SUBADDRESS;$0||$|REMOVE_SUBADDRESS;$1|
  *|$0* $Y

AUTH_REWRITE

! Check to see if the From: matches the authenticated user's mail attribute;
! exit with success if it does
  *|*|*|* $C$|MATCH_NOSUBADDRESS;$2$|$3|$E
! Fetch the base DN we'll need to look up the user
  *|*@*|*@* $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4
! Check to see if the From: matches the authenticated user's
! mailAlternateAddress, mailEquivalentAddress, or sendOnBehalfOf attributes.
  BASE|*|*|* \
  $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(|(mailAlternateAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(mailEquivalentAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(sendOnBehalfOf=$1)))[|$1|$2
! Exit with success if it does
  FOUND|*|*|* $E
! Now switch the base DN to that of the domain specified in the From: address
  BASE|*|*@*|*@* $CSECONDARY_BASE|$}$2,_base_dn_{|$1@$2|$3@$4
! Check and see if the address owner granted send-on-behalf-of permissions
  SECONDARY_BASE|*|*|* \
  $CSECONDARY_FOUND|$]ldap:///$0?uid?sub?(&(|(mail=$=$1$_)(mailAlternateAddress=$=$1$_)(mailEquivalentAddress=$=$1$_))(mailGrantSendPermissionsTo=$=$2$_))[|$2
```

```
! Exit with success if permission was granted - also add a Sender: field
  SECONDARY_FOUND|*|* $Y$1
! Fail any other From: address
  * $NYou$ are$ not$ authorized$ to$ send$ from$ the$ address$ you$ specified
```

In the case of a "mailGrantSendPermissionsTo" granting permission, this mapping also adds a Sender: field containing the authenticated address of the actual sender. This can be disabled by changing the "Y!" in the SECONDARY_FOUND check to "$E".

Note that this example retains the ability to specify a sendOnBehalfOf attribute on the user permissions are being granted to. If this is not desirable the "(sendOnBehalfOf=$1))" search clause should be removed from the associated LDAP URL.

Also note that neither of the two preceding examples grant permissions to "send on behalf of" using subaddresses; adding this capability is straightforward.

Finally, if permission checks on both sides must succeed - a logical AND rather than an OR - the following set of mappings could be used:

```
REMOVE_SUBADDRESS

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$0@$2
  "$_*+$_*"@*                           $Y"$0"@$2
  $_*+$_*@*                             $Y$0@$2
  *                                     $Y$0

REMOVE_SUB_LDAP_QUOTE

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*  $Y$=$0@$2$_
  "$_*+$_*"@*                           $Y$="$0"@$2$_
  $_*+$_*@*                             $Y$=$0@$2$_
  *                                     $Y$=$0$_

MATCH_NOSUBADDRESS

  *|*   $C$|REMOVE_SUBADDRESS;$0||$|REMOVE_SUBADDRESS;$1|
  *|$0* $Y

AUTH_REWRITE

! Check to see if the From: matches the authenticated user's mail attribute;
! exit with success if it does
  *|*|*|* $C$|MATCH_NOSUBADDRESS;$2$|$3|$E
! Fetch the base DN we'll need to look up the user
  *|*@*|*@* $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4
! Check to see if the From: matches the authenticated user's
! mailAlternateAddress, mailEquivalentAddress
  BASE|*|*|* \
  $CUSER|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(|(mailAlternateAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)(mailEquivalentAddress=$|REMOVE_SUB_LDAP_QUOTE;$1|)))[|$1|$2
! Exit with success if it does
  USER|*|*|* $E
! Must now match the sendOnBehalfOf attribute.
  BASE|*|*|* $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$=$2$_)(sendOnBehalfOf=$=$1$_))[|$1|$2
! Now switch the base DN to that of the domain specified in the From: address
  FOUND|*|*@*|*@* $CSECONDARY_BASE|$}$2,_base_dn_{|$1@$2|$3@$4
! Check and see if the address owner granted send-on-behalf-of permissions
  SECONDARY_BASE|*|*|* \
  $CSECONDARY_FOUND|$]ldap:///$0?uid?sub?(&(|(mail=$=$1$_)(mailAlternateAddress=$=$1$_)(mailEquivalentAddress=$=$1$_))(mailGrantSendPermissionsTo=$=$2$_))[|$2
! Exit with success if permission was granted - also add a Sender: field
  SECONDARY_FOUND|*|* $Y$1
! Fail any other From: address
  * $NYou$ are$ not$ authorized$ to$ send$ from$ the$ address$ you$ specified
```

46.3.30.4.2 Additional fields in AUTH_REWRITE probe (`authrewrite_extra_headers`)

New in MS 8.0.2.3. The `authrewrite_extra_headers` MTA option provides the means to include the content of additional header fields in `AUTH_REWRITE` mapping probes. The value consists of a space-separated list of header field names. Each field name specified creates an additional AUTH_REWRITE string argument at the end of the AUTH_REWRITE mapping probe, separated by vertical bars. If one or more of the specified field is present in the message header the value of the first such field will be included in the probe. Note that the separators are always present regardless of whether or not the associated field is part of the message.

All of the header field names must be known to the MTA; an error will be signalled if an unknown field name is specified. A colon, if specified at the end of a field name, will be ignored.

46.3.30.5 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 EXPN: command of:

```
530 5.7.0 Authentication required prior to EXPAND
```

while any attempted MAIL FROM: command will receive an error of:

```
520 5.7.0 Authentication required prior to MAIL/SAML/SEND/SOML
```

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. In legacy configuration, 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.) In Unified Configuration, those TCP/IP-channel-specific options have been replaced by the `authpassword`, `authusername`, and `externalidentity` channel options.

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 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 `saslpassauth` 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 `smtpauthuser` and `smtpauthpassword` MSHTTP options in Unified Configuration (or the `smtpauthuser` and `smtpauthpassword configutil` 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.)

Note that the `plaintextmincipher` MTA option, if set to a value greater than `0`, will restrict the user of plaintext passwords for authentication unless a security layer (SSL or TLS) is activated; see TLS and SASL channel options and Password and TLS MTA options for further discussion of SSL/TLS configuration for the MTA.

46.3.30.6 Automatic use of AUTH EXTERNAL at MAIL FROM (`explicitsaslexternal`, `implicitsaslexternal`)

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.

46.3.30.7 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 for the various protocols supported by the MTA, including but not limited to SMTP, LMTP, and MTQP.

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.

The ManageSieve server only supports the server subset of the TLS options since there is no ManageSieve client.

`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/MTQP 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/MTQP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP/LMTP/MTQP 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/MTQP client will attempt TLS use when sending outgoing messages, if sending to an SMTP/LMTP/MTQP server that supports TLS, and `maytlsserver`, which means that the SMTP/LMTP/MTQP 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/LMTP 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. This support is not available in MTQP.

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 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 neglible.)

Note that these options affect only TLS use negotiated at the 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.

46.3.30.8 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 accomodate 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.

46.3.30.9 XCLIENT SMTP Extension Support (`noxclient`, `xclient`, `xclientsasl`, `xclientrepeat`, `xclientsaslrepeat`)

(New in 8.0.) The MTA's SMTP server 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, PORT, DESTADDR, and DESTPORT 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|this-mta's-ip|25|1.2.3.4|12345
```

where note that the values to use in the "source IP" and "source port" fields have been specified via ADDR and PORT, respectively.

Note: Support for DESTADDR and DESTPORT was added in MS 8.0.2.3.

46.3.30.10 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 and AUTH_REWRITE mapping table, both of which can affect the MAIL FROM AUTH parameter.

46.3.30.11 saslruleset Option

RESTRICTED: Not yet implemented.

46.3.30.12 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 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 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.)

46.3.30.13 tlsmaxversion Option

The tlsmaxversion channel option determines the maximum acceptable version of TLS (the modern version of the SSL protocol). This presently takes a value of TLS1.0, TLS1.1, TLS1.2, orTLS1.3. This option defaults to TLS1.3.

46.4 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 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.

46.4.1 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.

46.4.2 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:

46.4.2.1 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.

46.4.2.2 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.

46.4.2.3 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.

46.4.2.4 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.

46.4.2.5 LINELENGTH (integer)

This option controls the length at which to fold headers. See also the discussion of the `headerlinelength` channel option.

46.4.2.6 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.

46.4.2.7 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.

46.4.2.8 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.

46.4.2.9 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 MS 6.3, the MTA's header line precedence processing was always applied. As of 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.)

46.4.2.10 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"
```

46.4.3 Header Fields Known to the MTA

The MTA maintains an internal list of "known" header fields. This list is used to perform and in some cases optimize various operations the MTA performs on message headers. In particular, the MTA's header trimming facilities are built on top of this list and only the specific fields on this list can be individually manipulated using header trimming. (The header addition facility, on the other hand, does not depend on this list in the same way and can be used to add arbitrary headers.)

The header fields currently on the list are:

```
A1-format:
A1-forward:
A1-function:
A1-type:
Accept-Language:
Action:
Address:
Addresses-referred-to:
Alternate-recipient:
App-message-id:
Apparently-to:
Application-name:
Approved-by:
Approved:
Archived-at: (added in MS 8.0)
Arrival-date:
Article-creation-date:
Attachment-date:
Attachment-encoding:
Attachment-import:
Attachment-name:
Attachment-type:
Attachment:
Authentication-results: (added in MS 7u3)
Authors:
Auto-forwarded:
Auto-submitted:
Autoforwarded:
Autosubmitted:
Bcc:
Beak:
Bilateral-Info:
Btw:
```

```
Cc:
Certificate:
Change-history:
Change-id:
Comments:
Complete-subject:
Content-access-control:
Content-alternative: (added in iMS 6.0)
Content-annotation:
Content-charset:
Content-class:
Content-comments:
Content-compression:
Content-correlator:
Content-creation-date:
Content-creator:
Content-description:
Content-disposition:
Content-encoding:
Content-features:
Content-future-object-size:
Content-id:
Content-identifier:
Content-label:
Content-language:
Content-last-modification-date:
Content-last-modifier:
Content-last-read-date:
Content-last-reader:
Content-legal-qualifications:
Content-length:
Content-lines:
Content-machine:
Content-MD5:
Content-mode:
Content-notification:
Content-object-size:
Content-operating-system:
Content-pathname:
Content-permitted-actions:
Content-privacy:
Content-related-stored-file:
Content-return:
Content-service:
Content-storage-account:
Content-transfer-encoding:
Content-type:
Conversation-id:
Conversion-with-loss:
Conversion:
Copies-to:
Data-description:
Data-encoding:
```

```
Data-name:
Data-type:
Date-delivered:
Date-posted:
Date-received:
Date-transferred:
Date-warning:
Date:
Deferred-delivery:
Defragment-failed:
DEK-info:
Deliver-by-date: (added in MS 8.0.1.0)
Delivered-to:
Delivery-date:
Delivery-receipt-to:
Delivery-report-content-billing-information:
Delivery-report-content-intermediate-trace:
Delivery-report-content-original:
Delivery-report-content-reported-recipient-info:
Delivery-report-content-UA-Content-id:
Destination-application:
Diagnostic-code:
Discarded-X400-IPMS-Extensions:
Discarded-X400-MTS-Extensions:
Disclose-recipients:
Disposition-notification-options:
Disposition-notification-to:
Disposition:
Distribution:
DKIM-Signature: (added in MS 7u1)
DL-expanded:
DL-expansion-history:
DL-expansion:
DomainKey-Signature: (added in MS 7u1)
DSN-gateway:
Encoding-info:
Encoding:
Encrypted:
Envelope-to: (added in MS 8.0.2.2)
Error-report:
Error: (added in iMS 6.0)
Errors-to:
Expiration-date:
Expires:
Expiry-date:
Exported-to:
Failure: (added in iMS 6.0)
Fake-sender:
Fcc:
Final-log-id:
Final-MTA:
Final-MTS-type:
Final-recipient:
```

```
Final-status:
Followup-to:
Form-type:
Forwarded-Message-ID: (added in MS 8.0.2.3)
Forwarded:
From:
Fruit-of-the-day-warning:
Fruit-of-the-day:
Full-name:
Future-release-request: (added in MS 7.0)
Generate-delivery-report:
Header-encoding:
Hop-count:
HTTP-Proxy:
HTTP-User-Agent:
Importance:
In-reply-to:
Incomplete-copy:
Information-type:
iPlanet-SMTP-Warning: (added in iMS 6.0)
Issuer-certificate:
Key-info:
Keywords:
Language:
Last-attempt-date:
Latest-delivery-time:
Licensed-to:
Lines:
List-Archive:
List-Digest:
List-Help:
List-Host:
List-Id:
List-Info:
List-Owner:
List-Post:
List-Software:
List-Subscribe:
List-Unsubscribe:
Mac-fcreator:
Mac-ftype:
Mail-followup-to:
Mail-system-version:
Mailer:
Mailing-List: (added in JES MS 6.2)
MCB-options:
MCB-type:
MDN-gateway:
Media-accept-features: (added in iMS 6.0)
Message-context: (added in iMS 6.0)
Message-discard:
Message-encoding:
Message-hash: (added in JES MS 6.2)
```

```
Message-id:
Message-sequence:
Message-type:
MHS-id:
MIC-info:
MIME-version:
MMDF-Warning:
MMHS-Acp127-Message-Identifier: (added in MS 8.0.1.0)
MMHS-Codress-Message-Indicator: (added in MS 8.0.1.0)
MMHS-Copy-Precedence: (added in MS 8.0.1.0)
MMHS-Exempted-Address: (added in MS 8.0.1.0)
MMHS-Extended-Authorisation-Info: (added in MS 8.0.1.0)
MMHS-Handling-Instructions: (added in MS 8.0.1.0)
MMHS-Message-Instructions: (added in MS 8.0.1.0)
MMHS-Message-Type: (added in MS 8.0.1.0)
MMHS-Originator-PLAD: (added in MS 8.0.1.0)
MMHS-Originator-Reference: (added in MS 8.0.1.0)
MMHS-Other-Recipients-Indicator-CC: (added in MS 8.0.1.0)
MMHS-Other-Recipients-Indicator-To: (added in MS 8.0.1.0)
MMHS-Primary-Precedence: (added in MS 8.0.1.0)
MMHS-Subject-Indicator-Codes: (added in MS 8.0.1.0)
MR-Received:
MS-TNEF-Correlator: (added in MS 8.0.1.0)
Msg-class:
MT-Priority: (added in MS 8.0.1.0)
Net-attachment:
News-software:
Newsgroups:
Next-attachment:
NNTP-posting-date:
NNTP-posting-host:
Notification-correlator:
Obsoletes:
Office:
Openpgp: (added in MS 8.0.2.3)
Organization:
Original-content-disposition:
Original-content-type:
Original-encoded-information-types:
Original-encoding-types:
Original-envelope-id:
Original-From: (added in MS 8.0.1.0)
Original-message-id:
Original-MTS-type:
Original-recipient:
Original-recipients:
Original-to: (added in MS 8.0)
Originating-client:
Originator-and-DL-expansion-history:
Originator-certificate:
Originator-ID-asymmetric:
Originator-ID-symmetric:
Originator-info:
```

```
Originator-return-address:
Originator:
P1-content-type:
P1-message-id:
P1-recipient:
P2-originator:
Path:
Phone:
PMDF-SMTP-Warning: (added in iMS 6.0)
Posted:
Posting-date:
Posting-status:
PP-Warning:
Precedence:
Prefer-Language:
Preferred-Language:
Prev-Resent-date:
Prev-Resent-from:
Prev-Resent-to:
Prevent-nondelivery-report:
Priority:
Proc-type:
Re-sent-by:
Read-receipt-to:
Received-by:
Received-from-MTA:
Received-from:
Received-SPF: (added in MS 7u4)
Received:
Recipient-ID-asymmetric:
Recipient-ID-symmetric:
Recipient-ID:
Recipient-info:
Recipient-reassignment:
Recipient:
Redirection-history:
References:
Refused-by:
Registered-mail-reply-requested-by:
Rejected-for:
Remote-MTA:
Remote-MTS-type:
Remote-recipient:
Remote-status:
Reply-by:
Reply-copies-to:
Reply-to:
Reporting-DL-name:
Reporting-MTA:
Reporting-UA:
Repository:
Requested-delivery-method:
Require-Recipient-Valid-Since: (added in MS 8.0)
```

```
Resent-bcc:
Resent-cc:
Resent-date:
Resent-from:
Resent-message-id:
Resent-reply-to:
Resent-sender:
Resent-to:
Resent-wide-reply-to:
Respond-by:
Ret-message:
Return-path:
Return-receipt-requested:
Return-receipt-to:
Send-to:
Sender-ID:
Sender:
Sensitivity:
Service-message:
Session-id:
Signature:
SMF-version:
Solicitation: (added in JES MS 6.2)
Source-info:
Spam-test: (added in iMS 6.0)
Status:
Structure:
Subject-submission-identifier:
Subject:
Summary:
Sun-Java-System-SMTP-Warning: (added in MS 7.0)
Sun-ONE-SMTP-Warning: (added in iMS 6.0)
Supersedes:
Telefax:
Text-type:
Thread-index:
Thread-topic:
To:
Total-copies-to:
Total-to:
Transport-options:
UA-content-id:
UID:
UIDL:
User-Agent: (added in iMS 6.0)
Version:
Via-host:
VMS-FDL:
Wanted-X400-conversion:
Warning: (added in iMS 6.0)
Warnings-to:
Wide-Reply-to:
Will-retry-until:
```

```
X-10:
X-11:
X-12:
X-1:
X-2:
X-3:
X-4:
X-5:
X-6:
X-7:
X-8:
X-9:
X-Accept-Language:
X-Address:
X-Admin:
X-Advertisement:
X-attachments:
X-Authentication-warning:
X-Authentication:
X-Author-info:
X-BCC:
X-BeenThere:
X-BFT:
X-BTW:
X-Buckslip:
X-CC:
X-Certificate:
X-Charset:
X-CLREnv-To: (added in iMS 6.0)
X-Comment:
X-Complaints-to:
X-Confirm-reading-to:
X-Content-description:
X-Content-disposition:
X-Content-id:
X-Content-transfer-encoding:
X-Content-type:
X-Corrupt-Content:
X-DEK-info:
X-Die-Spammers:
X-Dispatcher:
X-DMW-Body-names:
X-eGroups-Return: (added in JES MS 6.2)
X-Envelope-from:
X-Envelope-to:
X-EPUB-ListID:
X-EPUB-MsgID:
X-EPUB-SubID:
X-Eric-conspiracy:
X-Exchange-Antispam-Report-CFA-Test: (added in MS 8.0.2.2)
X-Exchange-Antispam-Report-Test: (added in MS 8.0.2.2)
X-EXP32-SerialNo:
X-Expiredinmiddle:
```

```
X-Face:
X-Favorite-drink:
X-Favourite-drink:
X-FAX-defaults:
X-FAX-number:
X-FAX:
X-Finfo:
X-Forefront-Antispam-Report: (added in MS 8.0.2.2)
X-Forwarded-for: (added in JES MS 6.2)
X-Gateway-source-info:
X-GBUdb-Analysis: (added in MS 8.0.1.0)
X-Genie-from:
X-Genie-id:
X-Gm-Message-State: (added in MS 8.0)
X-Google-DKIM-Signature: (added in MS 8.0)
X-Hop-count:
X-HPDesk-priority:
X-Incognito-format:
X-Incognito-SN:
X-Info:
X-Invoice: (added in MS 8.0)
X-Issuer-certificate:
X-Job: (added in MS 8.0.2.3)
X-Juno-Line-Breaks:
X-Key-info:
X-Keywords:
X-Licensed-to:
X-List-Archive:
X-List-Digest:
X-List-Help:
X-List-Host:
X-List-Id:
X-List-Info:
X-List-Owner:
X-List-Post:
X-List-Software:
X-List-Subscribe:
X-List-Unsubscribe:
X-List:
X-Listname:
X-Listprocessor-Version:
X-Listserver:
X-Listservers:
X-Loop:
X-Lotus-FromDomain:
X-LSV-ListID:
X-Machine:
X-Mailer:
X-Mailing-List:
X-Mailman-Version:
X-Message-flag: (added in iMS 6.0)
X-MessageSniffer-Clean: (added in MS 8.0.1.0)
X-MessageSniffer-Identifier: (added in MS 8.0.1.0)
```

```
X-MessageSniffer-License: (added in MS 8.0.1.0)
X-MessageSniffer-RulebaseUTC: (added in MS 8.0.1.0)
X-MessageSniffer-Rules: (added in MS 8.0.1.0)
X-MessageSniffer-Scan-Result: (added in MS 8.0.1.0)
X-MessageSniffer-SNF-Group: (added in MS 8.0.1.0)
X-MessageSniffer-Spam: (added in MS 8.0.1.0)
X-MessageSniffer-Version: (added in MS 8.0.1.0)
X-MessageSniffer-White: (added in MS 8.0.1.0)
X-MIC-info:
X-Microsoft-Exchange-Diagnostics: (added in MS 8.0.2.2)
X-MIME-Autoconverted:
X-MIME-version:
X-MIMEOLE:
X-MIMETrack:
X-Mms-3GPP-MMS-Version: (added in MS 8.0)
X-Mms-Ack-Request: (added in MS 8.0)
X-Mms-Delivery-Report: (added in MS 8.0)
X-Mms-Expiry: (added in MS 8.0)
X-Mms-Forward-Counter: (added in MS 8.0)
X-Mms-Message-Class: (added in MS 8.0)
X-Mms-Message-ID: (added in MS 8.0)
X-Mms-Message-Type: (added in MS 8.0)
X-Mms-MM-Status-Code: (added in MS 8.0)
X-Mms-Originator-System: (added in MS 8.0)
X-Mms-Previously-sent-by: (added in MS 8.0)
X-Mms-Previously-sent-date-and-time: (added in MS 8.0)
X-Mms-Priority: (added in MS 8.0)
X-Mms-Protocol: (added in JES MS 6.2)
X-Mms-Read-Reply: (added in MS 8.0)
X-Mms-Request-Status-Code: (added in MS 8.0)
X-Mms-Sender-Visibility: (added in MS 8.0)
X-Mms-Status-text: (added in MS 8.0)
X-Mms-Transaction-ID: (added in JES MS 6.2)
X-MS-Attachment:
X-MS-Embedded-Report:
X-MS-Exchange-CrossTenant-OriginalArrivalTime: (added in MS 8.0.2.2)
X-MS-Exchange-Organization-Journal-Report: (added in MS 7u4)
X-MS-Has-Attach: (added in MS 8.0.1.0)
X-MS-Journal-Report: (added in MS 7u4)
X-MSMail-conversation-id:
X-MSMail-message-id:
X-MSMail-priority:
X-MSXMTId:
X-MTS-LoopDetect:
X-MTS-Priority:
X-MTS:
X-MyDeja-info:
X-Netmessenger-type:
X-News-software:
X-Newsgroups:
X-Newsreader:
X-No-Archive:
X-Notes-form:
```

```
X-Notes-item:
X-NS-transfer-id:
X-NVL-Content-charset:
X-NVL-Content-filename:
X-NVL-Content-modification-date:
X-NVL-Content-transfer-encoding:
X-NVL-Content-type:
X-OldDate:
X-Open-Mail-hops:
X-Orcl-application:
X-Orcl-content-type:
X-Org-addr:
X-Org-misc:
X-Organisation:
X-Organization:
X-Orig-sender:
X-Original-From: (added in MS 8.0.1.0)
X-Original-MessageID: (added in MS 8.0.1.0)
X-Original-To: (added in MS 8.0.1.0)
X-OriginalArrivalTime:
X-Originating-IP:
X-OriginatorOrg: (added in MS 8.0.2.2)
X-Perlmx-Spam: (added in JES MS 6.2)
X-PGP-signed:
X-PGP-version:
X-Phone-number:
X-Phone:
X-PipeGCOS:
X-Pipehub:
X-Pipeuser:
X-PMrqc:
X-PMuue:
X-Priority:
X-Proc-type:
X-Proofpoint-Spam-Details: (added in MS 8.0.1.0)
X-Proofpoint-Virus-Version: (added in MS 8.0.1.0)
X-PS-Qualifiers:
X-Received:
X-Recipient-ID:
X-Reply-to:
X-Report-type:
X-Reposting-Policy:
X-Resent-bcc:
X-Resent-cc:
X-Resent-date:
X-Resent-from:
X-Resent-message-id:
X-Resent-to:
X-RPost-ClientCode: (added in MS 8.0)
X-RPost-Convert-CleanMetadata: (added in MS 8.0)
X-RPost-Convert-Pdf-Doc: (added in MS 8.0)
X-RPost-Convert-Pdf-Password: (added in MS 8.0)
X-RPost-Convert-Pdf-Ppt: (added in MS 8.0)
```

```
X-RPost-Convert-Pdf-Xls: (added in MS 8.0)
X-RPost-Convert-Pdf: (added in MS 8.0)
X-RPost-Convert-Zip: (added in MS 8.0)
X-RPost-Distrib: (added in MS 8.0)
X-RPost-Esign-Expiration: (added in MS 8.0)
X-RPost-Esign-Sequential: (added in MS 8.0)
X-RPost-Esign-Text: (added in MS 8.0)
X-RPost-Esign: (added in MS 8.0)
X-RPost-Language: (added in MS 8.0)
X-RPost-NoAck: (added in MS 8.0)
X-RPost-ReceiptCopy: (added in MS 8.0)
X-RPost-ReplyRegistered: (added in MS 8.0)
X-RPost-Seal-Hash: (added in MS 8.0)
X-RPost-Seal: (added in MS 8.0)
X-RPost-SecuRmail-AutoPassword: (added in MS 8.0)
X-RPost-SecuRmail-Password: (added in MS 8.0)
X-RPost-SecuRmail: (added in MS 8.0)
X-RPost-Sidenote-Bcc: (added in MS 8.0)
X-RPost-Sidenote-Cc: (added in MS 8.0)
X-RPost-Sidenote-Text: (added in MS 8.0)
X-RPost-Type: (added in MS 8.0)
X-Save-Headers:
X-Save-Outgoing:
X-Scanner:
X-Security:
X-Sender-ID:
X-Sender-IP:
X-Sender:
X-Sent-Mail:
X-Server-Date:
X-SMAP-Received-from:
X-SMTP-Client:
X-Source-IP: (added in MS 8.0.1.0)
X-SPAM-BAYESIAN__TOKEN: (added in JES MS 6.2)
X-SPAM-BDY-QUOTED_EMAIL_TEXT: (added in JES MS 6.2)
X-Spam-Flag: (added in MS 8.0)
X-SPAM-HDR-IN_REP_TO: (added in JES MS 6.2)
X-SPAM-HDR-REFERENCES: (added in JES MS 6.2)
X-SPAM-HDR-X_ACCEPT_LANG: (added in JES MS 6.2)
X-Spam-Level: (added in iMS 6.0)
X-SPAM-META-REPLY_WITH_QUOTES: (added in JES MS 6.2)
X-Spam-Score: (added in MS 8.0)
X-Spam-Status: (added in iMS 6.0)
X-Spook:
X-Status:
X-Sun-Charset:
X-Sun-Content-encoding:
X-Sun-Content-label:
X-Sun-Content-length:
X-Sun-Content-lines:
X-Sun-Data-description:
X-Sun-Data-name:
X-Sun-Data-type:
```

```
X-Sun-Encoding-info:
X-Sun-Text-type:
X-Sybari-Space:
X-To:
X-Trace:
X-Truth:
X-UID:
X-UIDL:
X-URI: (added in iMS 6.0)
X-URL:
X-Virus-Scanned: (added in MS 8.0.1.0)
X-VMS-Cc:
X-VMS-From:
X-VMS-To:
X-WebMail-Urgent:
X-WebMail-UserId:
X-WM-Posted-At:
X-X-Sender:
X-Yow:
X400-Content-correlator:
X400-Content-identifier:
X400-Content-return:
X400-Content-type:
X400-MTS-identifier:
X400-Originator:
X400-Received:
X400-Recipients:
X400-Trace:
Xref:
```

Chapter 47 Rewrite rules

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 editted "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.

[a]Technically, 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.

47.1 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 = 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 editting them "as if" they were the upper half of a legacy `imta.cnf` file.

47.2 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 jdoe %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.

Note [8]The 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.

47.2.1 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:

Table 47.1 Example Host Name Extractions During Rewriting

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 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.

47.2.2 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.

As of MS 8.0.2.2, if A-labels (see RFC 5890 section 2.3.2.1) are present they are both looked up as-is and in U-label form. Similarly, if U-labels are present they are converted to A-labels, looked up, then converted back to U-labels and looked up. Finally, any U-labels specified on the left hand side of a rewrite rule are converted to A-labels and back again when the rewrite rules are loaded, so they will match even if they are specified in uncanonical and/or unnormalized formats.

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.

As of MS 7.0U3, if the host/domain doesn't match and is a domain literal of the form "[channel:name]", where "name" is the name of an existing channel, a rewrite rule of the form:

```
[channel:name]          $U%$D@official-host-name
```

where "official-host-name" is the official channel host associated with the channel "name", is synthesized and used. This channel name form for domain names is primarily intended for use in source routes.

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 mtach.

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:

Table 47.2 Rewriting dan@sc.cs.cmu.edu Example

Pattern	Files Scanned

$*	configuration file/`rewrite` group rules and then domain database; this is the Initial match-all rule
sc.cs.cmu.edu	configuration file/`rewrite` group rules and then domain database
*.cs.cmu.edu	configuration file/`rewrite` group rules only
.cs.cmu.edu	configuration file/`rewrite` group rules and then domain database
..cmu.edu	configuration file/`rewrite` group rules only
.cmu.edu	configuration file/`rewrite` group rules and then domain database
..*.edu	configuration file/`rewrite` group rules only
.edu	configuration file/`rewrite` group rules and then domain database
sc.cs.cmu.edu	channel host name table (those official host names and host name aliases specified in channel definitions)
..*.*	configuration file/`rewrite` group rules only; this is an example of Rules to match domains containing exact numbers of components
.	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.

47.2.3 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 jdoe@domain.com has matched the rewrite rule

```
domain.com        $U@DOMAIN.COM
```

Then the template will produce the username jdoe, 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, `$U` = jdoe, `$H` = domain, and `$D` = .com. The template produces the username jdoe, the host/domain specification domain.com, and the routing system TCP-DAEMON.

47.2.4 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.

47.2.5 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.

47.2.6 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.

47.2.7 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.

47.3 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 jdoe@hosta.subnet.domain.com will match the specific hosta.subnet.domain.com rewrite rule pattern, while an address of jdoe@hostb.subnet.domain.com will match the more general .subnet.domain.com rewrite rule pattern, and an address of jdoe@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 47.3 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 rewritings, distinctions must be made in subsequent rewritings by controlling which rewrite rules match the address; see Tagged rewrite rule sets.

47.3.1 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
```

47.3.2 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.

47.3.3 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.

47.3.4 A rule to match any domain literal

If the MTA tries to rewrite an address whose domain is 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 `$E$R` 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.

47.3.5 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.

47.3.6 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 mispellings 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 to simplify 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!

47.3.7 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.

47.4 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.)

47.4.1 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 47.4 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.

47.4.1.1 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 "l" 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 "l" 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
```

47.4.1.2 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 detectedable 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`.

47.4.1.3 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 jdoe@host1.domain.com into the source routed address @hub.domain.com:jdoe@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.

47.4.2 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 `$U` is a substitution sequence. It causes the username portion of the address being rewritten to be substituted into the output of the template. Thus, if jdoe@host1.domain.com was being rewritten by this template, the resulting output would be jdoe@domain.com, the `$U` substituting in the username portion, jdoe, 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 47.5 Summary of template substitutions and control sequences

Substitution sequence	Substitutes
$D	Portion of domain specification that matched
$G	Insert current `default host`
$*n*G	Insert nth element, counting from left to right, of current `default host`
$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 and as of MS 8.1.0.6, temporary failures from routine callouts
$..	Turn off special handling of temporary failures; temporary failures will be 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 *n*
$]...[	LDAP search URL lookup
$(*text*)	General "database" substitution; rule fails if lookup fails
${...}	Apply specified mapping to supplied string
$n{...}	Apply specified mapping to supplied string along with prefixes specified by n.

$[...]	Invoke customer supplied routine; substitute in result
$&*n*	*n*th part of unmatched (or wildcarded) host as counting from left to right starting from 0
$!*n*	*n*th part of unmatched (wildcarded) host as counted from right to left starting from 0
$**n*	*n*th part of matching pattern as counting from left to right starting from 0
$#*n*	*n*th part of matching pattern as counted from right to left starting from 0
$*n*D	Portion of domain specification that matched, preserving from the *n*th leftmost part starting from 0
$*n*H	Portion of host/domain specification that didn't match, preserving from the *n*th leftmost part starting from 0
$Y	Equivalent to $1Y; 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
$*n*Y	Substitute in the *n*th (counting from the left, starting from 0) field of the transport information; hence, for incoming SMTP messages $0Y substitutes in the literal string " `TCP`", $1Y (see also $Y) substitutes in the SMTP server IP address, $2Y substitutes in the SMTP server port, $3Y substitutes in the SMTP client IP address, and $4Y substitutes in the SMTP client port
Control sequence	Effect on rewrite rule
$1~	Force channel match "success" effect, although truncating the rewrite rule at this point if the "real" channel match failed
$,	(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 `tlds.txt` file)
$>	(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 `tlds.txt` file)
$:	Apply only if the address being rewritten is the result of an alias
$;	Fail if the address being rewritten is the result of an alias
$E	Apply only to envelope addresses
$B	Apply only to header/body addresses
$F	Apply only to forward-directed (*e.g.,* To:) addresses
$R	Apply only to backwards-directed (*e.g.,* From:) addresses
$M*channel*	Apply only if channel *channel* is rewriting the address
$1M	Apply only if an "internal" channel is rewriting the address
$N*channel*	Fail if channel *channel* is rewriting the address
$1N	Fail if an "internal" channel is rewriting the address
$Q*channel*	Apply if sending to channel *channel*
$C*channel*	Fail if sending to channel *channel*
$S	Apply if host is from a source route
$A	Apply if host is to the right of the at sign

$P	Apply if host is to the right of a percent sign
$X	Apply if host is to the left of an exclamation point
$T*newtag*	Set the rewrite rule tag to *newtag*
$I*list-name*	Apply only if the *list-name* matches
$O*list-name*	Fail if the *list-name* matches
$?*errmsg*	If rewriting fails, return *errmsg* instead of the default error message
$*nxxxyyy*?*errmsg*	If rewriting fails, return SMTP error code *n.xxx.yyy* and error text *errmsg* instead of the default SMTP error code and error message
$V*domain*	Succeed if domainMap succeeds (that is, if the domain name *domain* is found in the LDAP directory)
$Z*domain*	Succeed if domainMap fails (that is, if the domain name *domain* is *not* found in the LDAP directory)
$*n*J	Set mailbox flags
$*n*K	Clear mailbox flags
$nT	Set override `alias_magic` value to be used for the domain which matched this rewrite rule when the rewrite rule is being used during alias expansion; *n* must be an appropriate `alias_magic` value
$n=B	(New in MS 8.0) Rewrite fails unless n is the message block size
$n>B	(New in MS 8.0) Rewrite fails unless n is greater than the message block size
$n>=B	(New in MS 8.0) Rewrite fails unless n is greater than or equal to the message block size
$n<B	(New in MS 8.0) Rewrite fails unless n is less than the message block size
$n<=B	(New in MS 8.0) Rewrite fails unless n is less than or equal to the message block size
$n<>B	(New in MS 8.0) Rewrite fails when n is the message block size
$n=L	(New in MS 8.0) Rewrite fails unless n is the number of lines in the message
$n>L	(New in MS 8.0) Rewrite fails unless n is greater than the number of lines in the message
$n>=L	(New in MS 8.0) Rewrite fails unless n is greater than or equal to the number of lines in the message
$n<L	(New in MS 8.0) Rewrite fails unless n is less than the number of lines in the message
$n<=L	(New in MS 8.0) Rewrite fails unless n is less than or equal to the number of lines in the message
$n<>L	(New in MS 8.0) Rewrite fails when n is the number of lines in the message
$n=P	(New in MS 8.0) Rewrite fails unless n is the message priority
$n>P	(New in MS 8.0) Rewrite fails unless n is greater than the message priority
$n>=P	(New in MS 8.0) Rewrite fails unless n is greater than or equal to the message priority
$n<P	(New in MS 8.0) Rewrite fails unless n is less than the message priority
$n<=P	(New in MS 8.0) Rewrite fails unless n is less than or equal to the message priority
$n<>P	(New in MS 8.0) Rewrite fails when n is the message priority

47.4.2.1 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 `subaddressexact` 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.

47.4.2.2 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. $*n*H and $*n*D 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 $*n*H and $*n*D 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.

47.4.2.3 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 47.6 Single field substitutions

Control Sequence	Usage
$&*n*	Substitute the *n*th element, *n*=0,1,2,..,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 *left* is element zero. The rewrite fails if the requested element does not exist.
$!*n*	Substitute the *n*th element, *n*=0,1,2,..,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 *right* is element zero. The rewrite fails if the requested element does not exist.
$**n*	Substitute the *n*th element, *n*=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the *left* is element zero. The rewrite fails if the requested element does not exist.
$#*n*	Substitute the *n*th element, *n*=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the *right* is element zero. The rewrite fails if the requested element does not exist.

Suppose the address 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.

47.4.2.4 Rewrite default host substitutions, $G, $*n*G

Any occurrences of `$G` are replaced with the current default host, as selected by the `defaulthost` and related channel option settings on the current source channel. Any occurrences of `$`*n*`G` are replaced with the value of the *n*th element, counting from zero, of the current default host; thus if the current default host has been determined to be `a.b.c.d.com`, `$0G` will substitute "`a`", `$1G` will substitute "`b`", *etc*. If no default host is set, *e.g.* through the use of `nodefaulthost`, then `$G` or `$`*n*`G` rewrite rules will fail (will not be considered to have matched).

47.4.2.5 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).

47.4.2.6 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.

47.4.2.7 Rewrite LDAP query URL substitutions, $]...[, $=

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 47.7 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 MS 6.0 through MS 6.2, and limited to 1024 characters as of 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 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.

47.4.2.8 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. Or new in MS 8.0, the general "database" can instead be stored in memcache; see the `general_database_url` MTA option. The on-disk database, if that is what is being used, is `IMTA_DATAROOT:db/generaldb` (which formerly could be redirected 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_TABLE: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.

47.4.2.9 Rewrite apply specified mapping substitutions, ${...}, $n{...}

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.

A substitution of the form $n{*mapping*,*argument*}, where n is a non-zero bit-encoded value works in same way, except that one or more |-separated prefixes are added to the mapping probe. The available prefixes are:

Table 47.8 Rewrite rule mapping probe prefixes

Bit	Value	Prefix
0	1	Transport information in the usual "TCP|*server-address*|*server-port*|*client-address*|*client-port*" format.
1	2	Application information.
2	4	Source channel (blank if no source channel is currently active).
3	8	Destination channel (blank if no destination channel is currently active).
4	16	Authenticated sender address (blank if no authenticated sender is currently active).

If multiple bits are set the corresponding prefixes are added in numerical order with the least significant bit on the left.

The ability to call out to a mapping can be used to make up for the lack of flexibility in rewrite rule matching - which sacrifices flexibility for speed. For example, a rewrite rule that matches and blocks attempts to send to DHCP client host names that take the form "dhcp-*.example.com" can be constructed using a mapping of the form:

```
DHCP_MATCH

  dhcp-*        $Y
```

And a rewrite rule of the form:

```
*.example.com  $E$F${DHCP_MATCH,$H}$?Cannot$ send$ to$ DHCP$ client
```

This mechanism also allows the MTA's rewriting process to be extended in complex ways. Substring matches on subdomains can be implemented and so can changes that cross subdomain boundaries. For example, suppose that domains of the form *-

subdomain.example.org are to be rewritten to *.subdomain.example.org in both header and envelope addresses and routed to the tcp_local channel. This can be done with a table:

```
DASH_TO_SUBDOMAIN

  *-subdomain    $Y$0.subdomain
```

And a rewrite rule of the form:

```
*.example.org    $U%${DASH_TO_SUBDOMAIN,$H}$D@TCP-DAEMON
```

The username part of an address can also be be selectively analyzed and modified, either alone or in conjunction with the domain part. For example, the following table and rewrite rule removes enclosing single quotes from all addresses, modifying both the local part and domain:

```
QUOTE_CHECK

  '*@*'          $Y$0%$1

$*               ${QUOTE_CHECK,$U$@$H}
```

47.4.2.10 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. As of MS 8.1.0.6, the special value 1112064044 may be returned to indicate a temporary failure.

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.

The site-supplied shared library image *image* should be readable but not writable by the Messaging Server user.

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.

47.4.2.11 Rewrite temporary failure handling, $.*text*., $..

By default, temporary LDAP failures and as of MS 8.1.0.6, routine callout failures, cause the current rewrite rule to fail. This is problematic in cases where different actions need to be taken depending on, say, 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. As of MS 8.1.0.6, this functionality has been extended to cover callout routine failures. 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 failures cause rewrite rule failure.

Note that all LDAP 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. This is in constrast to callout routines, where an explicit return value (1112064044) is used to indicate a temporary failure.

47.4.2.12 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.

47.4.2.13 Rewrite hash substitutions, $<...>, $n<...>

$<*text*> substitutes a hash of *text*. $*n*<*text*> substitutes a hash of *text*, modulo *n*.

47.4.2.14 Rewrite transport substitutions, $Y, $*n*Y

$*n*Y 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 tranport information, and bit 3/value 8 caused inclusion of tranport 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 `$Y` and `$`*`n`*`Y` is used in a rewrite rule template but transport information is not available for the address being rewritten, the rewrite rule will fail.

47.4.2.15 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.

`$M`*`channel`* causes the rule to fail if the channel *`channel`* is not currently doing the rewriting. `$N`*`channel`* 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.

47.4.2.16 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.

`$Q`*`channel`* causes the rule to fail if the channel *`channel`* is not the destination. `$C`*`channel`* 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.

47.4.2.17 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.

47.4.2.18 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.

47.4.2.19 Deployment map role-specific rewrites, $/, $|

As of the MS 8.0.2 release, it is possible to have rewrite rules whose application is dependent on the role of a host in the current deployment map.

Deployment map role rewrite checking is triggered by the presence of a $/ or $| control sequence in the template part of a rule. The characters following the $C or $Q, up until either an at sign, @, percent sign, %, subsequent channel, domain, or TLD check, or an error message $?, is should be of the form "*host*|*role*", where *host* is a host name and *role* is a role pattern.

A $/*channel* control sequence causes the rule to fail if the host *host* is not in the deployment map or it's role doesn't match the role pattern *role*. $|*host*|*role* causes the rule to fail if the role pattern matches the host's role in the deployment map.

Normally the *host* specification will itself be some sort of substitution, e.g. HD, $H, or $D. The exact form will depend on the desired effect as well as the form of the rewrite rule pattern.

The role pattern can contain glob-style wildcards.

Multiple $/ and $| clauses may be specified. If any one of multiple $/ clauses matches, the rule will succeed. If any of multiple $| clauses matches, the rule will fail.

47.4.2.20 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).

Note that the `imsimta test -domain_map` utility, and in particular its ENUMERATE command, can be used to probe/check/list domain definitions stored in LDAP.

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:l.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 47.9 Basic configuration settings relevant to domain LDAP lookups

msconfig base option	configutil parameter	MTA option	Default	Description
ugldaphost	local.ugldaphost	ldap_host	++	LDAP host to which to connect
ugldapport	local.ugldapport	ldap_port	389	LDAP port to which to connect
ldapsearchtimeout+	local.ldapsearchtimeout+	ldap_timeout	180000	Time out, in seconds, for LDAP queries
ugldapbinddn	local.ugldapbinddn	ldap_username		The username with which to bind when doing LDAP queries
ugldapbindcred	local.ugldapbindcred	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 (*e.g.*, due to ugldapusessl being set or use of an ldaps: URL), then enabling base.ldaprequiretls will require successful negotiation of TLS (using

				LDAP StartTLS) before proceeding with the connection.
		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
	local.imta.schematag	ldap_schematag	ims50	The tag for the schema in use
ldap_domain_ filter_schema1		ldap_domain_ filter_schema1	(|(objectclass= inetDomain) (objectclass= inetdomainalias))	Specifies the filter for domains when schema 1 is in use
ldap_domain_ filter_schema2		ldap_domain_ filter_schema2		Specifies the filter for domains when schema 2 is in use
ldap_domain_ known_attributes		ldap_domain_ known_attributes	-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
		domain_match_url		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 ldap:///$B?msgVanityDomain?sub?(msgVanityDomain=$D) if one wishes to support vanity domains
		domain_uplevel	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
		domain_failure	reprocess-daemon $Mtcp_local$1M $1~-error$4000000? Temporary lookup failure	What rewrite template to use if a $V or $Z rewrite rule lookup encounters an LDAP error, such as an LDAP connection error
ldap_domain_timeout		ldap_domain_timeout	900	Time (in seconds) to retain cached results of domain lookups (in the domain map library code cache)
		domain_match_ cache_size	100000	Number of domain lookup results to cache (in the MTA's cache)
		domain_match_ cache_timeout	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.

47.4.2.21 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 `$1~` 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 `$1~`.

47.4.2.22 TLD comparison rewrites, $, and $>

New in 7.0.5, the `$,` and `$>` control sequences allow checking domain top level portions against the current[1] 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.

[1] To keep the `tlds.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 `tlds.txt` file, issue the command `imsimta chbuild` to have the MTA rebuild its list.

As of MS 8.0.2.2, U-labels (see RFC 5890 section 2.3.2.1) are converted to A-labels before looking them up in tlds.txt. Note that the opposite is not true and specification of U-label domains in tlds.txt will not work.

47.4.2.23 alias_magic override rewrite, $*n*T

Rewrite rules can override the `alias_magic` MTA option setting and `aliasmagic` (source) channel option setting. Specifically, a construct of the form `$`*n*`T`, 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.

47.4.2.24 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 occured.

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.

47.4.2.25 List-name-sensitive rewrites, $I, $O

Mailing lists defined in the MTA can have amn 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.

47.4.2.26 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

$*N*comparison*C*

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.

47.4.2.27 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 channel whose official_host_nameTCP-DAEMON. This could be implemented with rules like the following (localhost is assumed to be the official name of the local host):

```
internet                $S$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 channel with `official_host_name` of `TCP-DAEMON`, regardless of any other criteria.

47.4.2.28 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@xyzzy.com
```

then any unrecognized host/domain specifications which will fail will, in the process of failing, generate the error message "Unrecognized address; contact postmaster@xyzzy.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.

47.5 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 48 Aliases

As part of the MTA's central role of routing messages, the MTA must recognize and potentially transform addresses to determine whither 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.

48.1 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 `$*` template (so that it matches all domains and will be consulted before all other rewrite rules---see Initial match-all rule), but with `$E $F` in the template so that it applies only to envelope To addresses (see Address direction and location-specific rewrites), then uses a pattern that uses `$V$H` 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` 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 `$E$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.

48.2 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.

Table 48.1 Basic configuration settings relevant to alias LDAP lookups

msconfig base option	configutil parameter	MTA option	Default	Description
`ugldaphost`	`local.ugldaphost`	`ldap_host`	++	LDAP host to which to connect
`ugldapport`	`local.ugldapport`	`ldap_port`	389	LDAP port to which to connect
`ldapconnecttimeout`	`local.ldapconnecttimeout`		10	Time out, in seconds, for LDAP connections
`ldapsearchtimeout+`	`local.ldapsearchtimeout+`	`ldap_timeout`	180000	Time out, in seconds, for LDAP queries
`ugldapbinddn`	`local.ugldapbinddn`	`ldap_username`		The username with which to bind when doing LDAP queries
`ugldapbindcred`	`local.ugldapbindcred`	`ldap_password`		The password with which to bind when doing LDAP queries
`ugldapusessl`	`local.ugldapusessl`		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 (extldap) lookups. If set to yes, then the MTA will expect to find a certificate located either in the `local.ssldbpath` directory, if it is specified, or in the MTA's config directory (located as `local.instancedir/config`, rather than via the usual IMTA_TABLE option value), named `local.ssldbprefix.secmod.db`. In that directory also must be the password file named `sslpassword.conf`.
`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 `ldaps:` URL), then enabling `base.ldaprequiretls` will require successful negotiation of TLS (using LDAP StartTLS) before proceeding with the connection.
Not used as of 7.0	`local.instancedir`			Prefix for where Messaging Server is installed. In particular, `/config` is appended onto this as the location for where to find the certificate (if SSL is being used; that is, if `local.ugldapusessl` is set).
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	*Varies with the schema tag*	The object classes required for a group; the default depends upon the schema tag; in particular, for the default schema tag of ims50, the default required object classes are inetLocalMailRecipient+inetmailgroup
		ldap_user_object_classes	*Varies with the schema tag*	The object classes required for a user; the default depends upon the schema tag; in particular, for the default schema tag of ims50, the default required object classes are inetLocalMailRecipient+inetmailuser
	local.imta.mailaliases	ldap_mail_aliases	*Varies with schema tag*	The attributes in which aliases are stored; the default depends upon the schema tag; in particular, for the default schema tag of ims50, the default alias attributes are mail, mailAlternateAddress, and mailEquivalentAddress
hostname	local.hostname	ldap_local_host		The local host name (official host name for the "l" channel)
	local.imta.hostnamealiases	ldap_host_alias_list		Local host aliases
ldappoolrefreshinterval	local.ldappoolrefreshinterval		35	Time (minutes) LDAP connections are maintained
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 pab.ldapport nor local.service.pab.ldapport is set, then the ldap_port 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.

48.3 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 editting aliases as if they were in the legacy configuration alias file; use `:q` to quit interactive editting mode, or `:x` to save your edits and exit interactive editting mode.

48.3.1 The alias group

In Unified Configuration, the `alias` group is not an option itself, but rather a grouping of alias setting defining a particular named alias.

48.4 Alias options

Alias options are the Unified Configuration equivalent of what in legacy configuration were termed Alias file named parameters, plus the alias translation value itself (specified by the `alias_entry` alias option). The alias options under each named `alias` group, together with the name, comprise the alias.

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.

48.4.1 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.*
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
```

48.4.2 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.

48.4.3 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.

48.4.4 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.

48.4.5 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.

48.4.6 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.

48.4.7 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.

48.4.8 Alias options: alias_autosecretary

RESTRICTED: Not yet implemented.

48.4.9 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.

48.4.10 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.

New in MS 8.0.1, see also the alias_capture_header alias option.

48.4.11 Alias options: alias_capture_header and alias_journal_header

(New in MS 8.0.1) The alias_capture_header alias option may be used to set an address to which to direct an encapsulated, "captured" copy of the message header of each message posted to the list. The alias_journal_header alias option works similarly, but generates an envelope "journal" format message header. (For full message content, not merely message header, in the captured copies, see the alias_capture and alias_journal alias options.) The *value* item should be the address to which to send the "captured" message copies. The alias_capture_header and alias_journal_header 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, when the LDAP attribute's value is tagged, respectively, ;format-report-header or ;format-journal-header.

The legacy configuration analogues are the [CAPTURE_HEADER] and [JOURNAL_HEADER] alias file named parameters.

48.4.12 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
```

then conversion 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 converter 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.

48.4.13 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 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-DD*THH:*MM*:*SS.ss*Z

or

*YYYY-MM-DD*THH:*MM*:*SS.ssplus-or-minus*HH:*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.

48.4.14 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 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.

48.4.15 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.

48.4.16 alias_description Option

The alias_description alias option takes a string argument and provides a way to associate a descriptive term or phrase with an alias. The value is unused otherwise.

48.4.17 Alias options: alias_digest_recurrence

RESTRICTED: Not yet fully implemented.

The alias_digest_recurrence alias option takes an ISO 8601 argument.

48.4.18 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.

48.4.19 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 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).

48.4.20 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.

48.4.21 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).

48.4.22 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.

48.4.23 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.

48.4.24 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/*>
```

48.4.25 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.

48.4.26 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.

48.4.27 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.

48.4.28 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.

48.4.29 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].

48.4.30 Alias options: alias_list_name

New in Messaging Server 7.4-0.01. RESTRICTED: Reserved for future use.

48.4.31 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_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.

48.4.32 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 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.

48.4.33 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].

48.4.34 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].

48.4.35 Alias options: alias_optin

The legacy configuration alias_optin alias option has been replaced in Unified Configuration by alias_optin1.

48.4.36 Spam/virus filter "opt in" alias options: alias_optinN (string)

The alias_optin*N* 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 [OPTIN*n*].

See also the ldap_optin*N* MTA options for similar functionality for aliases stored in LDAP.

48.4.37 Spam/virus filter "opt out" alias options: alias_optoutN (string)

The alias_optoutN alias options each respectively specify that the current alias is "opting out" of the corresponding spam/virus filter package N, for N in the range 1 to 8.

Note that the scope of opt-outs only extends to the immediate expansion values of the alias - if the alias expands to another alias that alias will not honor the outer level opt-out.

These alias options are analogues of the legacy configuration alias file named parameters [OPTOUTn].

See also the ldap_optinN MTA options for similar functionality for aliases stored in LDAP.

48.4.38 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.

48.4.39 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.

48.4.40 Alias options: alias_presence

RESTRICTED: Not yet implemented.

48.4.41 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.

48.4.42 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.

48.4.43 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 a 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].

48.4.44 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.

48.4.45 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].

48.4.46 Extra value alias options: alias_spare* (string)

The alias_spare*N* alias options (*N* = 1,...,18) are analogous to the attributes named by the ldap_spare_*N* MTA options, except with these options the corresponding spare slot is filled in when the alias is expanded. In legacy configuration, the analogous alias file named parameters are [SPARE*].

The spare value slots are 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.*).

48.4.47 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 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].

48.4.48 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].

48.4.49 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.

48.5 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
```

48.5.1 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 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 syntacticaly unexceptional) but in fact non-existant 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 report ed 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 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.

48.5.1.1 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.

48.5.1.2 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
```

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:

48.5.1.2.1 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.

48.5.1.2.2 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.

48.5.1.2.3 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 posting's 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.

48.5.1.2.4 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.

48.5.1.2.5 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.

48.5.1.2.6 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.

48.5.1.2.7 CAPTURE, JOURNAL

(CAPTURE is new in 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.

New in MS 8.0.1, see also the CAPTURE_HEADER and JOURNAL_HEADER named parameters.

48.5.1.2.8 CAPTURE_HEADER, JOURNAL_HEADER

(CAPTURE_HEADER and JOURNAL_HEADER are new in MS 8.0.1.) The CAPTURE_HEADER named parameter may be used to set an address to which to direct an encapsulated, "captured" copy of the message header of each message posted to the list. The JOURNAL_HEADER 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, when the LDAP attribute's value is tagged `;format-report-header` or `;format-journal-header`.

In Unified Configuration, see also the alias_capture_header and alias_journal_header alias options.

48.5.1.2.9 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 converter 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.

48.5.1.2.10 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-DD*T*HH*:*MM*:*SS*.*ss*Z

or

*YYYY-MM-DD*T*HH*:*MM*:*SS*.*ss**plus-or-minus**HH*:*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.

48.5.1.2.11 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 `deferreddestination` 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.

48.5.1.2.12 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`.

48.5.1.2.13 DIGEST_RECURRENCE

RESTRICTED: Not yet fully implemented.

The DIGEST_RECURRENCE parameter takes an ISO 8601 argument.

48.5.1.2.14 DIRECT_LIST, DIRECT_MAPPING

RESTRICTED: Not yet fully implemented.

48.5.1.2.15 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 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).

48.5.1.2.16 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.

48.5.1.2.17 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.

48.5.1.2.18 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.

48.5.1.2.19 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`.

48.5.1.2.20 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.

48.5.1.2.21 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.

48.5.1.2.22 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.

48.5.1.2.23 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.

48.5.1.2.24 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.

48.5.1.2.25 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.

48.5.1.2.26 LIST_NAME

New in Messaging Server 7.4-0.01; RESTRICTED.

48.5.1.2.27 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.

48.5.1.2.28 NOSOLICIT (comma-separated list of strings)

New in 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.

48.5.1.2.29 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_optinN` and `ldap_optoutN` MTA options.

48.5.1.2.30 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.

48.5.1.2.31 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.

48.5.1.2.32 PREFIX_TEXT, SUFFIX_TEXT

(New in 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.

48.5.1.2.33 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.

48.5.1.2.34 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.

48.5.1.2.35 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.

48.5.1.2.36 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.

48.5.1.2.37 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`.

48.5.1.2.38 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.

48.5.1.2.39 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*`.

48.5.1.2.40 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 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.

48.5.1.2.41 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.

48.5.1.2.42 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.

48.5.1.3 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.

48.5.1.3.1 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 part 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.

48.6 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.

48.6.1 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	File	Data base	File	Data base	File	Data base	Result
A	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*!

48.6.2 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.

Use the commands

```
# imsimta crdb input-file-spec IMTA_DATAROOT:db/aliasesdb-tmp
# imsimta renamedb IMTA_DATAROOT:db/aliasesdb-tmp IMTA_DATAROOT: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_DATAROOT:db/aliasesdb-tmp
# imsimta renamedb IMTA_DATAROOT:db/aliasesdb-tmp IMTA_DATAROOT:db/aliasesdb
```

48.7 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 "l" 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.

48.8 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.

48.9 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 48.2 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

48.10 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.

48.11 Alias restrictions

There are some important restrictions that should be observed when using aliases, especially aliases in the alias file or alias database.

48.11.1 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 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 MS 6.1, there is enhanced handling for such cases. Errors in e-mail addresses in group definitions (but not errors in LDAP DNs 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.

48.11.2 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.

48.11.3 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.

48.12 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.

48.12.1 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 `ugldapport` 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.

48.12.1.1 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 MS 6.2),
- `ldap_blocklimit` (as of MS 6.3) (normally `mailMsgMaxBlocks`),
- `ldap_source_channel` (as of MS 6.3),
- `ldap_source_optin`*`N`* (as of MS 6.3),
- `ldap_preferred_country` (as of MS 6.3), and
- `ldap_spare_`*`N`* (as of 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.

48.12.2 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_DATAROOT:db/reversedb-tmp
# imsimta renamedb IMTA_DATAROOT:db/reversedb-tmp IMTA_DATAROOT: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_DATAROOT: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_TABLE: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.

48.12.3 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 48.3 REVERSE mapping table flags

Flags	Description
$Y	Use output as new address
$N	Address remains unchanged
$D	Run output through the reversal database
$A	Add pattern as reverse database entry
$F	Add pattern as forward database entry
$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
$;C	(New in MS 7.0u1) Match only if this probe is not attempting to produce a canonical address for MTA use in comparison operations
$:D	(New in 8.0.1.3) Match only if this probe is attempting to produce a downgrade address for MTA use in EAI downgrade operations
$;D	(New in 8.0.1.3) Match only if this probe is not attempting to produce a downgrade address for MTA use in EAI downgrade 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.

48.12.4 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.

48.12.5 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 MS 6.2 for normal messages, or as of 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 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 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 48.4 PERSONAL_NAMES mapping table flags

Flags	Description
$Y	Use output as new personal name
$N	Personal name remains unchanged
Flag comparisons	Description
$:F	Match only forward pointing addresses
$;F	Match only if not a forward pointing address
$:R	Match only backwards pointing addresses
$;R	Match only if not a backwards pointing address
$:I	Match only message-ids
$;I	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 channnel 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 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 48.5 COMMENT_STRINGS mapping table flags

Flags	Description
$Y	Use output as new comment string
$N	Comment string remains unchanged
Flag comparisons	Description
$:F	Match only forward pointing addresses
$:R	Match only backwards pointing addresses
$:I	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.

48.13 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.

48.13.1 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`.

The questions of whether or not forwarded messages should be subject to "carryover" Sieve scripts and/or "opt in" to spam/virus filter package processing are controllable via the `sieve_user_carryoveroptin_user_carryover` MTA options.

48.13.2 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 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 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_spares2 MTA option control, respectively, the inclusion of MT-PRIORITY and expected message size, and LDAP "spare" attribute values associated with the sender address, 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, and the authenticated sender address, in the probe. With all these options enabled as well as the previously discussed options, the probe has the form:

src-chan|*from-addr*|*dst-chan*|*auth-sender*|*tag-list*|*s1*|*s2*|*s3*|*s4*|*s5*|*s6*|*mtpriority*|*expected-size*|*initial-to*|*inter-to*|*envelope-to*

New in MS 8.0.2.2, LDAP "spare" attribute values associated with the recipient address can also be included in the probe. These appear after the sender spare attributes and are controlled by additional bits in the include_spares2 MTA option.

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 48.6 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 YD 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 mgrpErrorsTo 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 at this alias level.
$I	Hold the message as a .HELD file.
$J	(New in 8.0.1.3) Disable further forward database or FORWARD mapping lookups for all inner levels. (Use with caution: Careless use of this option can prevent proper alias expansion from occurring.)
$K	(New in MS 7.0.4.27.5 and MS 7.0.5.29.0) Don't reset the "intermediate" address before processing the mapping/database result; only takes effect if $Y and $D are also set. This is useful when performing a final fixup to an address produced by delivery option processing.
$P	(New in MS 7.0.4.27.5 and MS 7.0.5.29.0) Treat the FORWARD mapping result as having specified additional recipient address(es) in addition to, rather than replacing, the current recipient address; only takes effect if $Y and $D are also set.
$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 *not* 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=SUCCESSES active for this recipient
$;S	(New in MS 6.3) NOTIFY=SUCCESSES 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
$:R	(New in MS 8.0) Match only if the current channel is "internal" (the process channel, reprocess channel, or conversion channel).
$;R	(New in MS 8.0) Match only if the current channel is *not* "internal".
$:U	(New in MS 8.0) Match only if a UTF8 address was used.
$;U	(New in MS 8.0) Match only if the address did not use UTF8.
$:V	(New in MS 8.0) Match only if the recipient address expanded via an alias.
$;V	(New in MS 8.0) Match only if the recipient address did not expand via an alias.
$:C	(New in MS 8.0.1.3) Match only if message is the result of a capture/journal action.
$;C	(New in MS 8.0.1.3) Match only if message is not the result of a capture/journal action.

See also the `ldap_domain_attr_catchall_mapping` MTA option which may be used to specify the name of a domain-level LDAP attribute whose value names an MTA mapping table that functions, for addresses in the domain which match no specific user entry, similarly to the `FORWARD` mapping table. That is, it provides a domain-specific, more restrictive (applying only to addresses which have no explicit user entry), analogue of the `FORWARD` mapping table.

48.13.3 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. Or new in MS 8.0, the forward "database" can be stored in memcache; seethe forward_database_url MTA option. 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-address changed-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 49 Mailing lists

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. Alias features fundamental to mailing list definitions include:

- Overriding the original envelope From address of each posting with the list "owner" address,
- A variety of flexible ways to specify the addresses of list members,

And further there are optional alias features that may be of particular interest on mailing list definitions, including:

- Size limits on posted messages,
- Posting restrictions (who may post to the list address), with many variants including:
  - Permission to post restricted to list members,
  - Permission to post requires authenticated message submission,
  - Permission to post granted to certain source domains,
  - Certain users or domains "black-listed" from posting,
  - Moderated lists (posting attempts require moderator review and approval),
  - List-specific password required to post,
- Deferred expansion (*i.e.*, "offline" expansion) of list membership,
- Application of a list-specific Sieve filter to attempted list postings,
- Addition of list-specific header lines,
- List-specific removal of header lines,
- "Tagging" of the Subject: header line of postings,
- Addition of prefix text or suffix text (*e.g.*, a "disclaimer") to list postings,

- List-specific conversion tag, possibly to trigger list-specific conversion operations on list postings,
- List-specific "opt in" to spam/virus filtering,

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 membership 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.

49.1 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 MTA LDAP attribute name options 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.

49.1.1 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).

49.1.2 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.

49.1.3 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 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";
   }
```

49.2 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 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 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.

49.2.1 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.

49.2.1.1 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
...
```

49.2.1.1.1 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
...
```

49.2.1.1.1.1 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 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
```

49.2.1.1.2 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 DNs, 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 DNs 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.

49.2.2 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 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 MS 6.1; instead such cases were silently ignored prior to 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 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 impliciations; 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 syntaces 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.

49.2.2.1 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.

49.2.3 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 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.

49.2.3.1 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_mapping*N* (*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_mapping*N* 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_mapping*N* 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.

49.2.4 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 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.

49.3 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 50 Mapping tables

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 `crdb` databases 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.)

50.1 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.

50.1.1 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 the 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, $; in 7.0.5 or later unquoted spaces and tabs are allowed in the second column.

A backslash, \, at the end of a line acts as a continuation character, causing the following line to be read and appended to the current line. A literal backslash may be placed at the end of a line by preceding it with a dollar sign, $\.

For clarity, it is recommended, but not required, that a blank line appear after each mapping table name and between each mapping table. Blank lines may also appear between successive mapping table entries but not after a continuation line, as it will terminate the continuation.

Trailing spaces and tabs are stripped from all lines, so lines containing nothing but spaces or tabs are considered "blank".

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 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 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).

When the MTA probes a mapping table, the overall probe length is also limited to 1024 characters.

The order in which mappings appear in the mappings file is not significant.

Duplicate mapping table names are not allowed in the mapping file.

50.1.1.1 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.

50.1.2 Mapping table format in Unified Configuration

In Unified Configuration, mapping tables may be editted 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
```

50.1.3 Mapping entry patterns

Mapping patterns can contain wildcard characters. In particular, the usual so-callled "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 50.1 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 *n*th wildcard or glob
Modifiers	Description
$_ ‡	Use minimal or "lazy" left-to-right matching for the following wildcard or glob
$@ ‡	Turn off "saving" for the following wildcard or glob
$^ ‡	Turn on "saving" for the following 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
$S or $S%	Match one symbol set character, *i.e.*, 0-9, A-Z, a-z, _, $
$S*	Match zero or more symbol set characters, *i.e.*, 0-9, A-Z, a-z, _, $
$T or $T%	Match one tab or vertical tab or space character
$T*	Match zero or more tab or vertical tab or space characters
$X or $X%	A synonym for $H%
$X*	A synonym for $H*
$[*c*] or $[*c*]%	Match character *c*
$[*c*]*	Match zero or more occurrences of character *c*
$[$c_1c_2...c_n$]	Match exactly one occurrence of character c_1, c_2, or c_n
$[$c_1c_2...c_n$]%	Match exactly one occurrence of character c_1, c_2, or c_n
$[$c_1c_2...c_n$]*	Match zero or more occurrences of any characters c_1, c_2, or c_n
$[$c_1$-$c_n$]	Match any one character in the range c_1 to c_n
$[$c_1$-$c_n$]%	Match any one character in the range c_1 to c_n
$[$c_1$-$c_n$]*	Match zero or more occurrences of characters in the range c_1 to c_n
$<*IPv4*>	Match an IPv4 address, ignoring bits
$(*IPv4*)	Match an IPv4 address, keeping prefix bits
${*IPv6*}	Match an IPv6 address

(‡) 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.

50.1.3.1 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.*

50.1.3.2 The $_ modifier: minimal *vs*. maximal matching

Asterisk, `*`, wildcards maximize what they match working from left to right across the input string. For instance, when the input string "`a/b/c`" is compared to the pattern `*/*`, the left asterisk will match "`a/b`" and the right asterisk will match "`c`".

The `$_` modifier causes wildcard matching to be minimized, where the least possible match is considered the match, working from left to right across the input string. For instance, when the input string "`a/b/c`" is compared to the pattern `$_*/$_*`, the left `$_*` will match "`a`" and the right `$_*` will match "`b/c`".

50.1.3.3 Controlling saving of wildcard or globs with $@ and $^

By default, the MTA keeps track of (saves) which character(s) are considered to have matched a wildcard or glob in a mapping table pattern for the duration of that mapping table entry (line). These retained character(s) can then be substituted back into the mapping table entry template via the `$0` through `$9` template substitutions. (That is, the character(s) matched in the left hand side (pattern) of a mapping table entry can be substituted back into the right hand side (template) of a mapping table entry, if desired.) But the MTA can only retain ten such matches. This restriction on the number of "saved" matches can always be dealt with by using multiple, iterative mapping table entries to achieve a desired effect (each entry doing only some matching/restoring, before leaving additional work to be done by a later entry). However, in many cases, where there is a desire to enforce a great deal of matching structure (more than ten wildcards or globs) but no need to reuse hence "save" the matched value(s), it is simpler to just "turn off" the saving via the `$@` modifier.

The `$^` modifier (the default behavior) turns on "saving". The `$@` modifier turns off saving. Note that such modifiers, when combined with a pattern wildcard glob or backmatch that already includes a dollar sign, `$`, in a syntax `$`*`w`*, are then indicated simply by including the modifier letter (but not the dollar sign) "within" the wildcard or back match syntax, as `$`*`mw`*, where *`m`* is the modifier character, and *`w`* is the wildcard or back match -- only one dollar sign character is used. For example, the syntax to match without saving any number of binary digits is `$@B*`, *not* `$@$B*`.

For instance, if it is desired to match fifteen of any sort of character:

```
%%%%%%%%%%$@%$@%$@%$@%$@%      $Y
```

matches and saves the first ten characters, and then matches without saving another five characters (though in fact in this example since *none* of the matched characters are actually substituted back into the template result, one could just as well have not bothered to save any of the matched characters -- that is, used the $@ modifier on all of the % character matches).

Or if it is desired to match eight alphabetic characters followed by a period followed by eight decimal digits, and retain only the first four alphabetic characters and the first four decimal digits, that could be done as:

```
$A$A$A$A$@A$@A$@A$@A.$D$D$D$D$@D$@D$@D$@D    $Y$0$1$2$3.$4$5$6$7
```

where here only the first four alphabetic characters and first four decimal digits are saved (and then substituted back into the template).

50.1.3.4 IP matching

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.

50.1.4 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 50.2 Mapping template substitutions and metacharacters

Substitution sequence	Substitutes
$*n*	*n*th wildcarded field as counted from left to right starting from 0. n must be a single digit
$'*n*'	*n*th wildcarded field as counted from left to right starting from 0. n can contain multiple digits
$%...%	(New in MS 6.2.) Substitute in a load average value.
$#...#	Sequence number substitution.
$+n#...#	Hash substitution.
$&...&	(New in 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.
$]$]$]...[	LDAP search URL lookup that concatenates multiple returned values separated by CRLFs.
$.*text*.	(New in MS 6.3.) Result to use in the case of temporary failures from LDAP lookups, domain attribute lookups, and routine callouts.
$}...{	LDAP domain map attribute lookup; substitute in attribute value.
${...}	General database substitution.
$|...|	Apply specified mapping table to supplied string.
$+n|...|	Apply specified mapping table to supplied string with one or more prefixes specified by the bit-encoded value *n*.
$`...'	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.
$+1C	(New in MS 8.1.0.6.) End processing of the current template now (unlike $C) and continue processing with the next mapping entry.
$E	End the mapping process now; use the output string from this entry as the final result of the mapping process.
$+1E	(New in 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 mapping table; use the output string of this entry as the new input string for the mapping process.
$+1R	New in MS 8.0.2.2. Continue the mapping process by repeating the current entry of the mapping table; use the output string of this entry as the new input string for the mapping process.
$?*x*?	Mapping entry succeeds *x* percent of the time.
$?*a,b,c...*?	Select a random value from a list of values. New in MS 8.0.2.3
$\	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.

50.1.4.1 Wildcard field substitutions, $n, $'n'

A dollar sign followed by a single digit *n* or one or more digits enclosed in single quotes *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.

50.1.4.2 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.

50.1.4.3 Processing control, $C, $+1C, $L, $R, $+1R, $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 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.

New in MS 8.1.0.6, the $+1C metacharacter sequence works like $C except processing of the current template is aborted and the remainder of the template is ignored. Note that this is likely to only be useful in conjunction with temporary failure handling and $..

New in MS 8.0.2.2, the $+1R metacharacter sequence provides a variant of $R: Rather than restarting the entire mapping from the first entry, it repeats the current entry with updated output from the current operation.

The $+1R is useful in constructing mappings that perform multiple replacements of all occurrences of one string with another where some of the strings overlap. A single operation of this sort can be done with $R; for example, replacing all percent signs with at-signs can be done with:

```
PERCENT_TO_AT

   *%*          $R$0@$1
   *            $0$Y
```

But when multiple replacements are involved restarting the mapping from the beginning is inefficient. And if the strings being replaced overlap restarting from the beginning won't work because the earlier, completed operations will see the later replacements and get confused. For this to work repeating the same entry until it no longer matches is required.

For example, the following mapping performs the quoting necessary for a string being inserted into an LDAP DN:

```
DN_QUOTE

   *\*          $+1R$0~$1
   *~*          $+1R$0\5c$1
   #*           $C\23$0
   *"*          $+1R$0\22$1
   *+*          $+1R$0\2b$1
   *,*          $+1R$0\2c$1
   *;*          $+1R$0\3b$1
   *<*          $+1R$0\3c$1
   *>*          $+1R$0\3e$1
   *            $0$Y
```

Note that if $R was used that the inital entry would damage the later replacement operations.

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.

50.1.4.4 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.

50.1.4.5 Entry randomly succeeds or fails, $?*x*?

A $?*x*? sequence 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 eighty 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|l|*    $C$?20?$Y
  tcp_*|busybee@some.where|l|*    $N$X4.5.9|Try$ again$ later
```

50.1.4.6 Load average substitutions, #%...%

At the present time load average substitutions are not implemented.

50.1.4.7 Select random entry from list, $?*a,b,c...*?

As of MS 8.0.2.3, a $?*a,b,c,...*? sequence consisting of two or more comma-separated string values in a mapping table entry causes one of the string values to be selected at random and retained in the output.

50.1.4.8 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 uniquifier 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 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 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`

A sequence number file accessed via a mapping table must be world readable in order to operate properly. (In particular, the `noprivuser` user id, as specified in the `restricted.cnf` file,is used to access the file.)

50.1.4.9 Hash substitutions, $+n#...#

(New in 7.4-18.01.)

A `$+n#...#` 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*#*string*#

or

$+*n*#*string*|*radix*#

or

$+*n*#*string*|*radix*|*width*#

$+*n*#*string*|*radix*|*width*|*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);
}
```

50.1.4.10 Character value substitutions, $&...&

(New in 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.

50.1.4.11 LDAP query URL substitutions, $]...[, $]$]...[, $]$]$]...[, $=

A substitution of the form `$]`*`ldap-url`*`[`, `$]$]`*`ldap-url`*`[`, or `$]$]$]`*`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 three forms is that in the case of a multi-valued result, `$]`*`ldap-url`*`[` uses the "first" returned value (note that the LDAP protocol leaves the order of values returned as unspecified and implementation-dependent -- thus you must not make any assumptions about which value might be returned "first"), whereas a multi-valued response is an error (the lookup is considered to have failed) for `$]$]`*`ldap-url`*`[`, and (new in 8.0) the `$]$]$]`*`ldap-url`*`[` form concatenates all the results together, separated by CRLFs.

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 permissable.) 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 `local.ugldaphost` and `local.ugldapport`). That is, the LDAP URL should be specified as:

```
ldap:///dn[?attributes[?scope?filter]]
```

or

```
ldap://host[: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. 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 superceded 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 MS 6.0 through MS 6.2, and limited to 1024 characters as of 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 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.

50.1.4.12 LDAP domain map attribute substitutions, $}...{

(New in MS 6.1p1/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 50.3 Domain map mapping template special syntaxes

Syntax	Interpretation and effect
`${`*`domain`*`,_base_dn_{`	Return the base DN for the user/group entries belonging to this domain
`${`*`domain`*`,_domain_dn_{`	Return the DN of the domain entry itself
`${`*`domain`*`,_domain_name_{`	Return the name of the domain (as opposed to an alias)
`${`*`domain`*`,_canonical_name_{`	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 50.4 User identifier mapping template attributes

Syntax	Interpretation and effect
`${`*`user-identifier`*`,#canonical_user#{`	Return the canonical form of user identifier
`${`*`user-identifier`*`,#canonical_domain#{`	Return the canonical domain the user is associated with
`${`*`user-identifier`*`,#user_dn#{`	Return the DN of the user's entry in the directory

(New in MS 8.0.2) A substitution of the form `${`*`host`*`,`*`query`*`{` is handled specially. *`host`* is treated as the name of a deployment map host.

The following queries can be specified:

Table 50.5 Deployment map mapping queries

Syntax	Interpretation and effect
`${`*`host`*`,/role/{`	return role of the specified host; fail if host does not exist in the deployment map
`${`*`host`*`,/online/{`	succeeds if the specified host exists and is online
`${`*`host`*`,/offline/{`	succeeds if the specified host is offline or does not exist
`${`*`user-identifier`*`,/<n>/{`	<n> must be an unsigned number - return the <n>th property value for the host; fail if host or specified property does not exist
`${`*`user-identifier`*`,/<p>/{`	<p> must be a string not matching any previous sort of query - succeeds if the specified host has properity <p> associated with it, fails otherwise. <p> may contain glob-style wildcards. On success the value of the first retained wildcard will be returned if it exists.

50.1.4.13 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. Or new in MS 8.0, the general "database" can instead be stored in memcache; see the general_database_url MTA option. The on-disk database, if that is what is being used, is IMTA_DATAROOT:db/generaldb (which formerly could be redirected 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 file IMTA_TABLE:general.txt (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.

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.

In some cases it may be useful to test to see if a given entry exists in the general database without substituting the corresponding template value from the database entry. This can be accomplished by creating a secondary mapping of the form:

```
GENERAL_EXISTS

   *                       $E${$0}$C
   *                       $Y
```

A mapping table substitution callout, described in the following section, with the general database key as the probe, will now succeed if and only if the corresponding entry is in the database, but will not return the corresponding template from the database entry.

50.1.4.14 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.

A substitution of the form $+n|*mapping*;*argument*|, where n is a non-zero bit-encoded value works in same way, except that one or more |-separated prefixes are added to the auxiliary mapping probe. The available prefixes are:

Table 50.6 Auxiliary mapping probe prefixes

Bit	Value	Prefix
0	1	Transport information in the usual "TCP|*server-address*|*server-port*|*client-address*|*client-port*" format.
1	2	Application information.
2	4	Source channel (blank if no source channel is currently active).
3	8	Destination channel (blank if no destination channel is currently active).
4	16	Authenticated sender address (blank if no authenticated sender is currently active).

If multiple bits are enabled the corresponding prefixes are added in numerical order with the least significant bit on the left.

Note that prefixes required an enqueue context; they will be ignored in mapping calls where such a context is not available.

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.

Mapping table substitutions have similarities to both subroutine calls and nesting of regular expressions using parentheses. The following recipient rate limit example illustrates the subroutine call aspect of mapping table substitutions by using a separate mapping table to remove any subaddress from an address so that subaddresses can't be used to avoid the limit.

A Metermaid throttle table with the following settings is used to implement a limit of 10 occurrence of a given recipient every 5 minutes:

```
msconfig> show metermaid.local_table:reclim
role.metermaid.local_table:reclim.data_type = string
role.metermaid.local_table:reclim.max_entries = 100000
role.metermaid.local_table:reclim.quota = 10
role.metermaid.local_table:reclim.quota_time = 300
role.metermaid.local_table:reclim.table_type = throttle
```

The following mapping removes any subaddress that's present in an address:

```
REMOVE_SUBADDRESS

  "$[a-z0-9.#$&'*\-/=?^_`{}\~]*+$_*"@*   $Y$0@$2
  "$_*+$_*"@*                            $Y"$0"@$2
  $_*+$_*@*                              $Y$0@$2
  *                                      $Y$0
```

And finally, the following callout rule added at the end of the ORIG_SEND_ACCESS mapping implements the limit:

```
tcp_*|*|*|* $C$;R$[IMTA_LIB:check_metermaid.so,throttle,reclim,$|REMOVE_SUBADDRESS;$3|]$X4.2.3|$NRate$ too$ high$E
```

Note the "$;R" (new in MS 8.0) prevents this rule from being applied to enqueues from "internal" channels (process channel, reprocess channel, conversions channel, etc.).

50.1.4.15 Expression substitutions, $`..'

In addition to basic arithmetic operations and tests listed in Operators in Order of Precedence, see Symbol table functions for a list of additional functions available for use in expressions. Note that mapping tables operate on strings; string arguments must be converted to integers (see for instance the `integer` function) before performing integer arithmetic on them, and conversely any arithmetic intermediate results must be converted back to strings (see for instance the `string` function) before being substituted back into mapping table entry templates.

Expression arithmetic can be used to manipulate IP addresses, which are a common component of many mapping probes. For example, the following mapping takes an IP address as input performs a general database lookup on successive subnets derived from the input address up to /8:

```
SUBNET_LOOKUP

  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$3/32}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>1<<1'/31}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>2<<2'/30}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>3<<3'/29}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>4<<4'/28}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>5<<5'/27}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>6<<6'/26}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2.$`$3>>7<<7'/25}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$2/24}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>1<<1'/23}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>2<<2'/22}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>3<<3'/21}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>4<<4'/20}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>5<<5'/19}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>6<<6'/18}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1.$`$2>>7<<7'/17}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$1/16}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>1<<1'/15}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>2<<2'/14}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>3<<3'/13}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>4<<4'/12}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>5<<5'/11}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>6<<6'/10}$Y$E
  $D*.$D*.$D*.$D*    $C${$0.$`$1>>7<<7'/9}$Y$E
  $D*.$D*.$D*.$D*    $C${$0/8}$Y$E
```

50.1.4.16 Mapping routine callout 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 in MS versions prior to 8.1.0.1. As of 8.1.0.2, the size has been increased to 1024 bytes. On UNIX systems **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. The special value 1112064044 may be returned to indicate a temporary failure.

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.

50.1.4.17 Temporary failure handling, $.text., $..

(New in 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 failure from an LDAP lookup, domain attribute lookup, or callout routine. By default these temporary failures cause the current mapping entry to fail, which is is problematic in cases where different actions need to be taken depending on, say, 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 failures cause mapping entry failure. Note that all LDAP and domain attribute 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. In contrast, callout routines have a unique return value (1112064044) used to indicate that a temporary failure has occurred.

50.2 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
```

50.3 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.

50.4 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` metacharacters, 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.

50.4.1 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, and Redis 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 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.

50.4.1.1 Database case sensitivity option (general_case)

New in MS 8.1.0.1. The general_case MTA option controls whether or not general database lookups are case sensitive. Setting the option to 1 makes the lookups case sensitive. The default is 0, in which case the lookups are case-insensitive.

50.5 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.

50.6 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_DEACCESS mapping table
- AUTH_REWRITE mapping table
- BURL_ACCESS mapping table
- CHARSET-CONVERSION mapping table
- COMMENT_STRINGS mapping table
- CONVERSIONS mapping table
- DEQUEUE_ACCESS mapping table
- DKIM_SIGN_DOMAINS mapping table
- DISPOSITION_LANGUAGE mapping table
- ETRN_ACCESS mapping table
- EXTERNAL_TO_INTERNAL 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_ACTIONS mapping table
- MILTER_MACROS mapping table
- MTA_STARTUP and MTA_SHUTDOWN mapping tables
- MX_ACCESS 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
- SASL_ACCESS mapping table
- SIEVE_EXTLISTS mapping table
- SMTP_ACTIONS 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.

50.7 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 tables 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.)

50.7.1 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]esmtpused`, `-[no]lmtpused`,

`-[no]proxyused`, `-[no]saslused`, `-[no]tlsused`. For instance, an `ORIG_SEND_ACCESS` mapping table of

```
ORIG_SEND_ACCESS

  tcp_local|friendly@somewhere.com|l|AdamUser@acme.com      $Y
  tcp_local|unwelcome@elsewhere.com|l|AdamUser@acme.com     $NGo$ away!
```

can be probed as follows:

```
# imsimta test -rewrite -from="friendly@somewhere.com" \
  -source=tcp_local -destination=l 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:
```

50.8 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.

See also the `mm_check_reputation` routine, discussed in Spamfilter early verdicts.

50.8.1 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)     (6)  (7)
```

The first four arguments common to all routines are:

flags (bit-encoded integer)
: 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 `check_memcache.so` *flags* parameter bit values below.

host (string)
: Host name of memcached server to use. The value specified by the `memcache_host` MTA option will be used if no host 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 50.7 `check_memcache.so` *flags* 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. This can be useful in meeting encrypted at rest requirements. The hash function used is controlled by the `memcache_hash_algorithm` MTA option.
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 preceeded 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. *quota_time* specifies the specifies the duration of the period over which counts are recorded, and *quota* specifies the maximum number of counts to permit during the period.

The available routines and their specific parameter formats are:

add,*flags*,[*host*:[*port*]],*key*,*value*,*timeoout*

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 retured 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 already 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 already returned.

`remove,flags,[host:[port]],key[,lockout]`

Remove the entry with the specified *key*. The *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 value 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 already 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 store in the entry. The call will fail if this check fails.

This callout has also been enhanced to work with the mapping table `$.` facility in order to be able to handle temporary errors in a sensible fashion.

An example of using memcached to implement a simple rate limit on a given authenticated sender to 10 messages every 5 minutes would be to add the following to the end of the FROM_ACCESS mapping:

```
FROM_ACCESS
  TCP|*|SMTP*|MAIL|tcp_*|*|*  $C$;R$[IMTA_LIB:check_memcache.so,throttle,0,memcache.example.com:22122,sendlim-$4,10,300]$X4.2.3|$NRate$ too$ high$E
```

This would limit users to 10 messages every 5 minutes (300 seconds). Here "memcache.example.com:22122" would be replaced with name and port of an actual memcached server.

A more flexible mechanism would allow the limit to be specified on a per-user basis along with a default. This could be accomplished by defining an LDAP attribute for the purpose, say

senderRateLimit, and making it available to the FROM_ACCESS mapping via the sixth spare LDAP slot:

```
msconfig> set ldap_spare_6 senderRateLimit
msconfig# set include_spares1 32
```

The follow additions to the FROM_ACCESS mapping would then be needed:

```
FROM_ACCESS

  TCP|*|SMTP*|MAIL|tcp_*|*|*|  $C$;R$[IMTA_LIB:check_memcache.so,throttle,0,memcache.example.com:22122,sendlim-$4,10,300]$X4.2.3|$NRate$ too$ high$E
  TCP|*|SMTP*|MAIL|tcp_*|*|*|*  $C$;R$[IMTA_LIB:check_memcache.so,throttle,0,memcache.example.com:22122,sendlim-$4,$5,300]$X4.2.3|$NRate$ too$ high$E
```

50.8.2 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.

An example of using Metermaid to implement a simple rate limit on a given authenticated sender to 10 messages every 5 minutes implemented using a FROM_ACCESS mapping would be:

```
msconfig> show metermaid.local_table:sendlim
role.metermaid.local_table:sendlim.data_type = string
role.metermaid.local_table:sendlim.max_entries = 100000
role.metermaid.local_table:sendlim.quota = 10
role.metermaid.local_table:sendlim.quota_time = 300
role.metermaid.local_table:sendlim.table_type = throttle
```

And finally, the following callout rule added at the end of the `FROM_ACCESS` mapping implements the limit:

```
FROM_ACCESS

  TCP|*|SMTP*|MAIL|tcp_*|*|*  $C$;R$[IMTA_LIB:check_metermaid.so,throttle,sendlim,$4]$X4.2.3|$NRate$ too$ high$E
```

Note the "$;R" (new in MS 8.0) prevents this rule from being applied to enqueues from "internal" channels (process channel, reprocess channel, conversions channel, etc.). Also note that the rule can be added to an existing

50.8.3 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 blacklist 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 several 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`
- `dns_verify_ipv4` (new in 8.0.1.2)
- `dns_verify_ipv6` (new in 8.0.1.2)
- `dns_get_first_mx` (new in 8.0.2.2)

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.

50.8.3.1 The `dns_verify` routine

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*`]]]`

domainname	The name to look up in the DNS and local host tables.
success (optional)	If specified, it is the mapping table string to return if *domainname* is found. If not specified, the default is "$Y".
`failure` (optional)	If specified, it is the mapping table string to return if *domainname* is not found. If not specified, the default is "$N".
unknown (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 *success* 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:

Table 50.8 dns_verify callout substitutions

String	Value
%a	If successful, the %a 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.
%e	If the lookup is not successful, %e substitutes the error message associated with the lookup.
%n	If successful, %n substitutes the primary name for *domainname*. 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
```

50.8.3.2 The dns_verify_ipv4 routine

The dns_verify_ipv4 routine is identical to dns_verify, except that it restricts its results to IPv4 addresses.

50.8.3.3 The dns_verify_ipv6 routine

The dns_verify_ipv4 routine is identical to dns_verify, except that it restricts its results to IPv6 addresses.

50.8.3.4 The dns_verify_ptr routine

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*]]]

ip-address	The IPv4/IPv6 address to be looked up, without any enclosing brackets or prefixes.
success (optional)	If specified, it is the mapping table string to return if *ip-address* is found. If not specified, the default is "$Y".
failure (optional)	If specified, it is the mapping table string to return if *ip-address* is not found. If not specified, the default is "$N".
unknown (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 *success* 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.

50.8.3.5 The dns_verify_domain and dns_verify_domain_port routines

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*]

ip-address	The IP address that you want to check against the blackhole list
domainname	The name of the blackhole list to check against, *e.g.*, blackholes.mail-abuse.org
text-string (optional)	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, either of dns_verify_domain or dns_verify_domain_port looks up the following name: 2.0.0.127.bl.spamcop.net. They 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
```

50.8.3.6 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:

ip-address,*domainname*[,*text-string*[,*header*]]

The *ip-address*, *domainname*, and *text-string* arguments are the same as for dns_verify_domain and dns_verify_domain_port. *header* is optional. If specified, it is a string containing the header name, and other optional text, to include before the TXT record string or *text-string* value. The header name must be one that the MTA recognizes. The default is "X-Dispatcher: ".

The following example shows an ORIG_MAIL_ACCESS mapping table entry to query spamcop.net:

```
ORIG_MAIL_ACCESS

 TCP|*|25|*|*|*|*|*|*|* \
$C$[IMTA_LIB:dns_verify,dns_verify_domain_warn,$1,\
bl.spamcop.net,spamcop.net:$ entry$ found$ for$ $$1,\
X-Dispatcher:$ SPAMfilter$ (spamcop.net):$ ]$E
```

For a source IP address of 127.0.0.2, this example would return

```
$Y$AX-Dispatcher: SPAMfilter (spamcop.net): Blocked - see http://spamcop.net/bl.shtml?127.0.0.2
```

This is then added as a header to the message. One way to act on this is to create a system-wide, channel, or user Sieve filter containing a Sieve action along the lines of:

```
if header :contains "X-Dispatcher" "SPAMfilter" { discard; }
```

50.8.3.7 dns_get_first_mx

New in MS 8.0.2.2. The `dns_get_first_mx` routine performs an MX record lookup on the specified domain. Unlike the other DNS routines described in this section, the routine argument consists solely of the domain to look up. The routine succeeds if the lookup is successful and returns the "first" MX record using the same algorithm the MTA uses to determine record order. An empty string is returned if the lookup succeeds but no MX records are found.

The routine fails if the specified domain cannot be found in the DNS. A temporary failure condition is returned if the DNS operation fails with a temporary error.

One possible use of this routine is to determine if a given domain is serviced by a particular collection of servers - which can then be used to implement a limited form of MX rollup. For example, a mapping that will determine if a given domain is handled by Office 365 servers could (at preent) be coded as:

```
OUTLOOK_MX

  *                          $E$[IMTA_LIB:dns_verify,dns_get_first_mx,$0]$C
  *.protection.outlook.com   $Y
```

This mapping could in turn be called from either a rewrite rule or a `FORWARD` to route messages to a special channel configured to deliver to an appropriate set of MXes regardless of destination domain.

50.8.4 smartsend callouts

The `smartsend.so` library provides a collection of mapping callouts that can be used in to optimize the delivery of opt-in bulk email.

smartsend makes use of a site-provided database that provides information about senders, deployment hosts, recipient domains, and available source IPs. The MTA's general database facility is used for this purpose, although note that this can easily be configured to perform queries using the memcache or Redis protocols.

Many smartsend options also require a cache server. smartsend talks to this server using either the memcache or Redis protocols. A per-host instance of memcached is sufficient for this purpose, although a few functions require deployment of a caching server available across multiple hosts. If the memcache protocol is employed the host and port specified by the `memcache_host` and `memcache_port` channel options respectively, with fallback to the corresponding MTA options, will be used.

When used to support multiple independent message sources, it is expected that the first conversion tag associated with each message will be used as a "virtual MTA" identifier. Note

that the callouts consume the conversion tag if it is present; they do not provide the means to set it.

The following sections describe the database format as well as the various callouts smartsend provides. It is important to understand that most of the callouts are expected to be configured as a group since they work together to provide a single coherent service. Most of the callouts are designed not to have any effect, i.e., return a failure condition, when they have nothing to do, so they may be combined with other uses of the corresponding mapping.

While the following sections provide details as to how to configure the smartsend callouts, this may be done automatically in a unified configuration by using the provided smartsend recipe:

```
# msconfig
msconfig> run smartsend.rcp [debug-level]
```

An optional debug level may be specifed; if none is specified a level of 0 (debugging disabled) is assumed. The recipe may be re-run with a different debug level; if this is done the debug level will be updated in the various callouts.

Although it's included as part of the smartsend plugin, the MX rollup capability is an essentially separate facility. Accordingly, a separate recipe is provided to configure it:

```
# msconfig
msconfig> run rollup.rcp [debug-level [rollup-domain-suffix [rollup-channel]]]
```

An optional debug level, rollup domain suffix, and the channel used to handle rollups may be specified. These default to "0", ".rollup", and "tcp_rollup", respectively.

The script will create the rollup channel if it does not already exist, using the tcp_local channel as a model.

50.8.4.1 Database entry formats

Database entries make use of several common pieces of syntax, the first of which is a parameter list. Parameter lists are represented using a simplified variant of MIME content-type parameter syntax. The ABNF (RFC 2234) syntax for a smartsend parameter list is:

```
parameter-list = parameter-name-value [";" parameter-name-value]
parameter-name-value = *WSP parameter-name *WSP "=" *WSP parameter-value *WSP
parameter-name = (ALPHA / DIGIT / "-" / "_")1*
parameter-value = <any CHAR excluding ";">
```

Note that the allowable characters in a parameter-value is often further constrained by the context in which the parameter-list appears. In particular, vertical bars ("|"), commas (","), and whitespace are all used in some "outer" contexts as delimiters.

Examples of parameter lists include:

```
maxmx=5
maxmessagerateperdomain=30/1200;maxmessagesperconnection=20
dkimidentity-1=example.net;dkimselector-1=brisbane
```

```
debug=3
```

A parameterized list of IP addresses (IP list for short) is also used by several types of database entries. The following definition makes use of the IPv4address and IPv6address productions specified in RFC 5954:

```
parameterized-ip-list = [global-parameter-list "|"] ip-and-parameters ["," ip-and-parameters]
global-parameter-list = [parameter-list]
ip-and-parameters = ["-"] (IPv4address / IPv6address) ["#" (IPv4address / IPv6address)] [";" ip-parameter-list]
ip-parameter-list = [parameter-list]
```

Note that IP address parameters allow the two address form supported by the `interfaceaddress` channel option.

Parameterized IP lists are allowed to be up to 4096 characters in length.

Examples of parameterized IP lists would include:

```
1.1.1,1,2.2.2.2,-3.3.3.3,4.4.4.4
10.59.230.40;banner_host=mauve.example.com,10.59.230.169;banner_host=plum.example.com
66.218.59.24#10.59.230.40;banner_host=mauve.example.com
maxconnectionrateperdomain=2/300|10.59.230.40,10.59.230.169
maxmessagesperconnection=30|10.59.230.40;maxmessagsperconnection=10,10.59.230.169,10.59.230.170
```

A parameterized list of MTA names (MTA list for short) is also used by several types of database entries. The following definition makes use of the Domain productions specified in RFC 5321:

```
parameterized-mta-list = [global-parameter-list "|"] mta-and-parameters ["," mta-and-parameters]
global-parameter-list = [parameter-list]
mta-and-parameters = ["-"] Domain [";" mta-parameter-list]
mta-parameter-list = [parameter-list]
```

Parameterized MTA lists are allowed to be up to 4096 characters in length.

Examples of parameterized MTA lists would include:

```
host1.example.com,host2.example.com,host2.example.com
host1.example.org;received_domain=host1.example.org,host2.example.org;received_domain=host2.example.org,
id_domain=example.edu|host1.example.edu,host2.example.edu
```

50.8.4.2 auth_access callout

The `auth_access` callout is intended to be called from the `AUTH_ACCESS` mapping. It provides the ability to:

- set various limits and operating parameters based on the message's destination domain
- select a client IP address from a group of available IP addresses for each message in a round-robin fashion
- impose additional limits and opeational settings on a per-IP basis.

The `AUTH_ACCESS` mapping should be set up as follows:

```
AUTH_ACCESS
```

```
*      $C$[IMTA_LIB:smartsend.so,auth_access,<pflags>|$0]$E
```

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output. Bit 3 (value 8), if set, enables routing-only mode, which is described below.

Note that the callout receives the entire mapping probe. It is designed to process the entire probe and automatically adjusts its behavior to match the variations in probe format caused by setting various MTA options.

Additionally, bit 0, value 1, of the `include_retries` MTA option should be set, and if conversion tags are to be used to select IP address groups, bit 11, value 2048, of the `include_conversiontag` MTA option should also be set. Note that the list of conversion tags is interpreted as follows:

```
<virtual-MTA>,<sender-OCID>,<tenant-OCID>,<compartment-OCID>
```

The callout normally operates as follows:

1. Parse the mapping probe into its components. The callout will fail if the input string cannot be parsed.
2. Construct a key of the form `domain_<domain>`, where <domain> is the lower cased destination domain for the message, ad look up the key in the general database. The value of any entry found is expected to be a parameter list.
3. If `<tenant-OCID>` conversion tag is present construct a key of the form `ociddomain_<tenant-OCIDA>_<domain>`, where <tenant-OCIDA> and <domain> are the lower cased tenant OCID and destination domain for the message, respectively, and look up the key in the general database. The value of any entry found is expected to be a parameter list.
4. If `<sender-OCID>` conversion tag is present construct a key of the form `ociddomain_<sender-OCIDA>_<domain>`, where <sender-OCIDA> and <domain> are the lower cased sender OCID and destination domain for the message, respectively, and look up the key in the general database. The value of any entry found is expected to be a parameter list.
5. Construct a key of the form `ips_<mtaid>_<virtual-MTA>`, where <mtaid> is the MTA id and <virtual-MTA> is the first conversion tag associated with the current message. Both the id and conversion tag are converted to lower case.

   If no MTA id has been specified the official host name on the local channel is used instead. The preceding "_" is omitted if the specified MTA id is blank.

   If there are no conversion tags associated with the message use a key of the form `ips_<mtaid>_<channel>`, where <channel> is the channel where the message is queued.
6. Look up the key in the general database. The value of any entry found is expected to be an IP list.
7. If an IP list entry is found select an IP from the list at a random starting point, and proceeding around the list in a strict round-robin fashion on subsequent dequeue attempts.

8. If an EXTERNAL_TO_INTERNAL mapping exists, apply it to any IP address selected from the IP list that just specifies an IP address (as opposed to an external#internal address pair). The entry will be skipped if the mapping doesn't produce a result and set $Y. The result can be either just the internal address, which will be become the internal IP address for the entry, or an external#internal pair, which will replace the IP address information in the entry.

9. Impose any limits and set any operational parameters specified by the various parameter lists attached to the destination domain, the IP list as a whole, and the chosen IP.

10. Limit checks may cause the selected IP addresses to be become ineligible for use. If this happens try the next IP address on the list. If all of the addresses are rejected the message dequeue attempt will fail with a temporary error, causing the MTA to try again later.

11. If no IP list entry is found and a conversion tag is present construct a third key of the form mta_<conversiontag> and look it up in the general database. Again, the conversion tag is converted to lower case.

12. If an MTA entry is found it is expected to consist of a comma or whitespace-separated list of MTA names that are able to process messages with the specified conversion tag. The list is checked to see if the current MTA is listed. If it isn't an MTA that is on the list is chosen at random and the message destination is set to that MTA.

13. If the current MTA is on the list or no MTA list is present the dequeue operation proceeds normally.

If bit 3 (value 8) of <pflags> is set, the plugin operates in routing-only mode:

1. Parse the mapping probe into its components. The callout will fail if the input string cannot be parsed or if no conversion tag is present

2. Construct a key of the form mta_<virtual-MTA> and look it up in the general database. The conversion tag is converted to lwoer case.

3. If an MTA entry is found it is expected to consist of a comma or whitespace-separated list of MTA names that are able to process messages with the specified conversion tag. The list is checked to see if the current MTA is listed. If it isn't an MTA that is on the list is chosen at random and the message destination is set to that MTA.

4. If the current MTA is on the list or no MTA list is present the dequeue operation proceeds normally.

Note that the addition of an IP address to a IP list entry when a message is being retried unavoidably results in the disruption of the strict round-robin schedule. However, a special mechanism is provided to remove an address from the rotation without causing any disruption: Placing a minus sign "-" in front of an IP address on the list will cause that address to be skipped. A count of the number of IP address skips is maintained in order to preserve the strict round-robin usage. Note that this does NOT work if the IP address is simply removed from the entry.

Also note that the log_smartsend MTA option may be used to include additional information about smartsend callout actions in transaction log records.

The following sections describe the parameters available for use in auth_access database entries. Note that while all parameters are available in all contexts, it may not make sense to,

say, specify the banner host on the basis of the destination domain or to control the use of TLS on a per-IP basis.

50.8.4.2.1 backoff - Retry frequency for messages

The backoff parameter implements the same functionality as the backoff channel option on smartsend domain entries. The syntax is identical to that of the backoff channel option in unified configuration.

50.8.4.2.2 banner_host - Override banner host

The banner_host parameter specifies the host name to use in the EHLO/HELO command that is issued once a connection is established. This overrides BANNER_HOST TCP/IP-channel-specific option for this connection.

50.8.4.2.3 chunking - Control use of SMTP CHUNKING

New in MS 8.1.0.1. The chunking parameter provides the ability to control the use of the SMTP CHUNKING extension by the SMTP client. Possible values are:

- "disable", which disables the use of chunking, as if nochunkingserver had been specified on the channel, and
- "optional", which allows the use of chunking if the channel is set to allows it.

50.8.4.2.4 debug - Enable smartsend/channel debugging

New in MS 8.1.0.1. The debug parameter provides the ability to enable debugging on a per-IP-list basis. A single integer argument is required. Positive values enable corresponding levels of smartsend debugging; master_debug is also enabled for the duration of processing the current message.

50.8.4.2.5 ipbackoff - Retry frequency for messages in IP backoff mode

The ipbackoff parameter implements the same functionality as the ipbackoff channel option on smartsend domain entries. The syntax is identical to that of the ipbackoff channel option in unified configuration.

50.8.4.2.6 ipbackofftimeout - Timeout for IP backoff entries

The ipbackofftimeout parameter implements the same functionality as the ipbackofftimeout channel option on smartsend domain entries. As with the channel option, the syntax is an integer value in seconds.

50.8.4.2.7 log_headers - Header fields to log in transaction record

The log_headers parameter is used to specify one or more additional header fields to log as part of the transaction log entry. The parameter value is a comma-separated list of header field names. Unlike header option file based logging, any field name may be specified, recognized or not. Note that the log_header MTA option must also be set appropriately.

50.8.4.2.8 maxconnectionrateperdomain - Maximum connection rate to a domain

The maxconnectionrateperdomain parameter provides the means to limit the rate at which connections are made to a destination domain. The value of this parameter is one or two

space-separated rate limit lists. The first limit list value specifies the rate limits to use when the IP address is operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

The syntax of rate limit lists is described in the section on message rate limits below.

50.8.4.2.9 `maxconnectionsperdomain` - Maximum connection to a domain

The `maxconnectionsperdomain` parameter implements limits on the number of simultaneous connections that can be opened from a single source IP to a destination domain. The value of this parameter is one or two space-separated connection limit lists. The first connection limit list value specifies the limits to use when the IP address is operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

As of MS 8.1, four types of per-domain connection limits are provided:

Table 50.9 Per-domain connection limits

Additional Qualification	Code	Probe	Description
Source IP	I	`maxconnectionsperdomain_`*<IP>*`_`*<domain>*	Per source IP per domain limit
Host	H	`maxconnectionsperdomain_`*<host>*`_`*<domain>*	Per server host per domain limit
Tag	T	`maxconnectionsperdomain_`*<tag>*`_`*<domain>*	Per conversion tag per domain limit
	N	`maxconnectionsperdomain_`*<domain>*	Per domain limit

Any or all of these limits may be engaged simultaneously. If multiple limits are used all limits must be satisfied for the connection attempt to proceed.

A connection limit list is specified as one or more limit codes followed by the limit itself, written as an unsigned integer. The "I" code may be omitted from a per-IP limit if it appears first - this is the only supported format in MS 8.0.2.3. Formally, the syntax is:

```
connection-limit-list = (ip-limit-value *limits) / 1*limits
limits = ip-limit / host-limit / tag-limit / limit
ip-limit = "I" ip-limit-value
ip-limit-value = 1*DIGIT
host-limit = "H" host-limit-value
host-limit-value = 1*DIGIT
tag-limit = "T" tag-limit-value
tag-limit-value = 1*DIGIT
limit = "N" limit-value
limit-value = 1*DIGIT
```

For example, a connection limit list value of "I4N10" specifies a 4 connection per-IP-per-domain limit and a 4 connection per-domain limit. A value of "10" specifies a 10 connection per-IP-per-domain limit.

Note that the identically named `maxconnectionsperdomain` channel option can be used to set connection limits on a per channel basis.

50.8.4.2.10 `maxmessagerateperdomain` - Maximum message rate to a domain

The `maxmessagerateperdomain` parameter provides the means to limit the rate at which messages are delivered to a destination domain. The value of this parameter is one or two space-separated rate limit lists. The first limit list value specifies the rate limits to use when the IP address is operating normally; the second specifies the limits to use when operating in IP backoff mode. If the IP backoff value is omitted it defaults to the first value.

As of MS 8.1, four types of per-domain rate limits are provided:

Table 50.10 Per-domain message rate limits

Additional Qualification	Code	Probe	Description
Source IP	I	`maxmessagerateperdomain_`*<IP>*`_`*<domain>*	Per source IP per domain limit
Host	H	`maxmessagerateperdomain_`*<host>*`_`*<domain>*	Per server host per domain limit
Tag	T	`maxmessagerateperdomain_`*<tag>*`_`*<domain>*	Per conversion tag per domain limit
	N	`maxmessagerateperdomain_`*<domain>*	Per domain limit

Any or all of these limits may be engaged simultaneously. If multiple limits are used all limits must be satisfied before a delivery attempt will be made.

A rate limit list is specified as one or more limit codes followed by the limit itself, written as an unsigned vulgar fraction or integer. The "I" code may be omitted from a per-IP limit if it appears first - this is the only supported format in MS 8.0.2.3. The numerator of the vulgar fraction specifies the number of messages to allow and the denominator specifies the time window in seconds. If a single integer is specified it is treated as a numerator with a default denominator of 3600 (one hour).

Formally, the syntax is:

```
rate-limit-list = (ip-limit-value *limits) / 1*limits
limits = ip-limit / host-limit / tag-limit / limit
ip-limit = "I" ip-limit-value
ip-limit-value = 1*DIGIT
host-limit = "H" host-limit-value
host-limit-value = 1*DIGIT
tag-limit = "T" tag-limit-value
tag-limit-value = 1*DIGIT
limit = "N" limit-value
limit-value = 1*DIGIT ["/" 1*DIGIT]
```

For example, a rate limit list value of "I100N100000/86400" specifies a 100 messages per hour per-IP-per-domain limit and a 100,000 per day per-domain limit. A value of "50" specifies a 50 messages per hour per-IP-per-domain limit. A value of "0" indicates that no rate limit is set.

Note that the identically named maxmessagerateperdomain channel option can be used to set rate limits on a per channel basis.

50.8.4.2.11 maxmessagesperconnection - Maximum messages per connection

The maxmessagesperconnection parameter limits the number of messages that can be transferred by a single connection. It overrides the ATTEMPT_TRANSACTIONS_PER_SESSION TCP/IP-channel-specific option for this connection.

50.8.4.2.12 max_mx_records - Maximum MX attempts

The max_mx_records parameter limits the number of MX records to consider when attempting to connect. It overrides MAX_MX_RECORDS TCP/IP-channel-specific option for this connection.

50.8.4.2.13 override_host - Override destination host

The override_host parameter specifies the host name to use as the destination for connections, overriding any host name that appears in any destination address.

50.8.4.2.14 status - Force hold, return of messages

The status parameter provides the means to return or hold messages. Possible values are "active", which causes deliveries to proceed normally, "hold", which suspends deliveries, and "return", which causes messages to be returned with no further delviery attempts.

50.8.4.2.15 tls - Control use of TLS

New in MS 8.1.0.1. The tls parameter provides the ability to control the use of the SMTP CHUNKING extension by the SMTP client. Possible values are:

- "disable"- disables the use of TLS, as if notlsserver had been specified on the channel,
- "optional", which allows the use of tLS if the channel is set to allows it, and
- "require", which requires the use of TLS, as if musttlsserver had been specified on the channel,

50.8.4.3 auth_deaccess callout

The auth_deaccess callout is intended to be called from the AUTH_DEACCESS mapping. It pairs with the auth_access callout described in the previous section to decrement counters associated with some of the limits auth_access implements. At present this is the only functionality provided by this callout.

The AUTH_DEACCESS mapping should be set up as follows:

```
AUTH_DEACCESS

   *     $C$[IMTA_LIB:smartsend.so,auth_deaccess,<pflags>|$0]$E
```

Here <pflags> should be replaced with an unsigned integer flag value. Currently only the lowest three bits of this bit-encoded value have any meaning -- they enable increasing levels of debug output, with "0" resulting in no output.

Note that the callout receives the entire mapping probe. It is designed to process the entire probe and adjusts its behavior to match the variations in probe format caused by setting various MTA options.

50.8.4.4 conversions callout

The `conversions` callout is intended to be called from the `CONVERSIONS` mapping to provide a database-driven means of configuring multiple DKIM signing keys.

The `CONVERSIONS` mapping should be set up as follows:

```
CONVERSIONS

   *     $C$[IMTA_LIB:smartsend.so,conversions,pflags=<pflags>;$0]$E
```

IMPORTANT NOTE: The format of the mapping probe is NOT the same as for the previously AUTH_ACCESS or AUTH_DEACCESS mapping probes. Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits of this bit-encoded value enable increasing levels of debug output, with "0" resulting in no output.

Note that the callout receives the entire mapping probe. It is designed to process the entire probe and adjusts its behavior to match the variations in probe format caused by setting various MTA options.

Additionally, bit 0, value 1, of the `include_domain` MTA option should be set, and if conversion tags are to be used to select DKIM keys using "dkim_" lookups. bit 1, value 2, of the `include_conversiontag` MTA option should also be set. IMPORTANT NOTE: "dkim_" lookups should be disabled if "ocid_" lookups in the AUTH_REWRITE mapping are enabled for DKIM use. In this case bit 4 (value 16) of pflags should be set to disable such lookups.

The callout operates as follows:

1. Parse the mapping probe into its components. The callout will fail if the input string cannot be parsed.
2. Construct a key of the form `domain_<domain>`, where <domain> is the lower cased destination domain (DOMAIN value in the CONVERSIONS mapping probe) for the message.
3. Look up the key in the general database. The value of any entry found is expected to be a parameter list.
4. If bit 4 (value 16) of the pflags value is clear, construct a second key of the form `dkim_<conversiontag>`, where <conversiontag> is the first conversion tag (TAG value in the CONVERSIONS mapping probe) associated with the current message and converted to lower case. If there are no conversion tags associated with the message use a key of the form `ips_<channel>`, where <channel> is the destination channel (OUT-CHAN value in the CONVERSIONS mapping probe) for the message.
5. Look up the key in the general database. The value of any entry found is expected to be another parameter list.
6. Use the combined parameter lists to select appropriate DKIM keys to use to sign the message, and generate a mapping result that will cause this to happen.

7. If bit 3 (value 8) of the pflags value is set and a conversion tag is present, construct a key of the form `mta_<conversiontag>` and look up the corresponding entry in the general database. If an entry is found it is expected to consist of a list of the MTAs capable of handling this conversion tag value. If an entry is found and the current MTA is not on the list the is forced into the tcp_intranet channel queue so it can be routed to the correct MTA.

50.8.4.4.1 `dkimidentity-N`, `dkimselector-N` - DKIM parameters

The actual DKIM parameters correspond exactly to the DKIM template keywords (dkimidentity-N and dkimselector-N) used in the CONVERSIONS mapping table and have the same semantics as the `destinationdkimidentityN` channel option. Note, however, that the parameter separator is a semicolon while the CONVERSIONS template keyword separator is a comma.

50.8.4.5 ip_backoff callout

New in MS 8.1. The `ip_backoff` callout is used to activate or deactivate IP backoff mode for a specified IP address and destination domain. Unlike other smartsend callouts, the `ip_backoff` callout is not associated with any specific mapping. Rather, it is designed to be used as part of site-specific checks that determine when IP backoff mode needs to be in effect.

A call to the `ip_backoff` callout should appear as follows:

```
$[IMTA_LIB:smartsend.so,ip_backoff,pflags=<pflags>|<channel>|<ip>|<domain>|<value>|<timeout>|]
```

The arguments are:

<pflags> Unsigned integer flag value. Currently only the lowest three bits of this bit-encoded value have any meaning -- they enable increasing levels of debug output, with "0" resulting in no output.

<channel> Channel associated with the IP address.

<ip> IP address for which IP backoff is to be enabled or disabled.

<domain> Destination domain for which backoff is to be enabled or disabled.

<valuet> The backoff entry's value. At the present time the entry value is ignored and this field should be left blank - but see the special value and their semantics below.

<timeout> Timeout, in seconds, for the bckoff entry. A value of 0 causes the timeout specified by the `ipbackofftimeout` channel option to be used. If no timeout value has been specified a default value of 3600 (one hour) is used. The timeout argument can be omitted, in which case it's value defaults to 0.

The callout succeeds even if the entry cannot be set.

The special value "?" causes a lookup to be performed on the backoff entry instead of a set. The callout fails if the entry is not present. If the entry is present the callout succeeds and returns the entry's value.

The specil value "-" causes the specified backoff entry to be deleted. The callout always succeeds.

The ip_backoff callout operates by creating and deleting entries in memcache or Redis. The entry's name is of the form "ip_backoff_<ip>" and at present the value is the string "mode=1".

50.8.4.6 auth_rewrite callout

The auth_rewrite callout is intended to be called from the AUTH_REWRITE mapping. It provides the ability to confirm that a conversion tag appearing in a header field has a corresponding IP list or MTA entry, the ability to activate DKIM signing, and can be used to override the received_domain and id_domain settings for the current message.

The confirmation functionality is useful in preventing messages from being routed to the wrong MTA during configuration updates. The ability to override domain settings can be used to obfuscate local host information.

If DKIM support and obfuscation support is not enabled the callout succeeds if the confirmation attempt fails, so that the mapping template itself determines the appropriate action to be taken. If, on the other hand, DKIM or obfuscation support is enabled, the callout succeeds when it needs to return a result and fails otherwise, in which case the plugin necessarily provides all results.

Note that this callout is only needed in specific circumstances and therefore is not configured by the smartsend.rcp recipe.

The AUTH_REWRITE mapping should be set up approximately as follows if bit 4 (value 16) of pflags is clear:

```
AUTH_REWRITE

    *       $C$[IMTA_LIB:smartsend.so,auth_rewrite,<pflags>|$0]$E$N$X4.7.0|Unknown$ tag
```

Note that the template in this example causes a temporary failure to be returned should confirmation fail.

The callout should be configured as follows if either bit 4 (value 16, DKIM) or bit 5 (value 32, domain obfuscation) are set:

```
AUTH_REWRITE

    *       $C$[IMTA_LIB:smartsend.so,auth_rewrite,<pflags>|$0]$E
```

Note that the template in this example simply returns the callout result if the callout succeeds.

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output. Bit 3 (value 8), controls the type of lookup that is performed for confirmation and domain obfuscation information. A lookup for "ips_" entries is performed if the bit is clear; a lookup for "mta_" entries is performed if the bit is set. If bit 5 (value 32) is set the result may contain override values for received_domain and/or id_domain.

Bit 4 (value 16) or bit 5 (value 32), if set, cause a lookup to be performed on a key of the form ocid_<sender-OCID>, where <sender-OCID> is the sender OCID (see below) If this lookup

fails an additional query is done for `ocid_<tenant-OCID>`, where <tenantr-OCID> is the tenant OCID. The result is interpreted as a parameter list and can contain DKIM parameters (bit 4 set) and override values for `received_domain` and/or `id_domain` (bit 5 set).

Additionally, the `authrewrite` channel option should be set to 16 on all appropriate source channels (or on the defaults channel if the mapping is to be applied to all message flows). The header where the virtual MTA (first conversion tag) is located may be specified as the first value of the `authrewrite_extra_headers` MTA option. The header field containing the tenant and sender OCIDs may be specified as the second value if DKIM processing is enabled. At present the only supported syntax for this header field is a JSON string of the form:

```
{"tenantId":"<tenant-OCID>","senderId":"<sender-OCID>","compartmentId":"<compartment-OCID>"}
```

The tenantId and senderId OCIDs must be present. The compartmentId is optional; if it is not supplied its value defaults to that of the tenantId. Note that the parameters may appear in any order. Any additional parameters will be silently ignored.

Alternately, bit 13, value 8192 of the `include_conversiontag` MTA option, can be set to cause conversion tag information to be included in the probe if it is available. The conversion tag list is interpreted as follows:

```
<virtual-MTA>,<sender-OCID>,<tenant-OCID>,<compartment-OCID>
```

The compartment-OCID will be omitted if it has the same value as the tenant-OCID, which is the case when it specifies the root compartment for the tenancy.

50.8.4.7 send_access callout

The `send_access` callout is intended to be called from either the `SEND_ACCESS` or `ORIG_SEND_ACCESS` mapping. It provides the ability to confirm that a conversion tag transferred using the XCONVTAG extension has a corresponding IP list or MTA entry. This functionality is useful in preventing messages from being routed to the wrong MTA during configuration updates. The callout succeeds if the confirmation attempt fails, so that the mapping template itself determines the appropriate action to be taken.

Note that this callout is only needed in specific circumstances and therefore is not configured by the smartsend.rcp recipe.

The `SEND_ACCESS` mapping should be set up approximately as follows:

```
SEND_ACCESS

    *      $C$[IMTA_LIB:smartsend.so,send_access,<pflags>|$0]$E$N$X4.7.0|Unknown$ tag
```

Note that the template in this example causes a temporary failure to be returned when confirmation fails.

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output. Bit 3 (value 8), controls the type of lookup that is performed. A lookup for "ips_" entries is performed if the bit is clear; a lookup for "mta_" entries is performed if the bit is set. (The latter is appropriate for an intermediate routing MTA.) Finally, bit 4 (value 16) should be set if this callout is placed

in the ORIG_SEND_ACCESS mapping (as opposed to its normal location in the SEND_ACCESS mapping).

Additionally, bit 4, value 16, of the include_conversiontag MTA option should also be set so that the conversion tag is included in the SEND_ACCESS mapping probe. Bit 3, value 8 should be set if the callout is done from the ORIG_SEND_ACCESS mapping.

50.8.4.8 forward callout

The forward callout is used in conjunction with the mx_access callout (described below) to implement a general purpose MX rollup facility that's superior to what was provided by the dns_get_first_mx mapping callout. With this approach rollups are specified in the general database, which in turn can be linked to either memcache or Redis.

The first step in configuring this functionality is to select a domain suffice to identify all of the rollups that will be defined. ".rollup" is used in other products for this purpose, but any domain can be used as long as it isn't a valid top-level domain (TLD) or internal domain.

Once the domain has been selected a rewrite rule needs to be added to route rollups to the proper tcp_ that will process rolled up messages. Assuming that ".rollup" is the rollup domain suffix, the tcp_local channel is to be used, and the channel has the usual official host name of "tcp_local-daemon", the rule would be:

```
.rollup                         $U%$H$D@tcp_local-daemon
```

The forward callout should be now added to the FORWARD mapping table as follows:

```
FORWARD

   *|*.rollup $E
   *          $C$[IMTA_LIB:smartsend.so,forward,<pflags>|$0]$E
```

Of course ".rollup" should be replaced with whatever domain is to be used to identify rollups.

The mx_access callout must also be set up, as described below, and the entries need to be added to the general database to create specific MX rollups.

The forward callout performs the following actions:

- Perform an MX record lookup on the routing domain part of the envelope recipient address. No action is taken if the lookup fails or no records are found.
- Construct a list of MX records with the lowest precedence (highest priority).
- For each MX host on the list, construct a probe of the form rolluphost_<mxhost> and look it up in the general database. The MX host name is forced to lower case as part of constructing the probe. The values of these entries are expected to either be of the form:

  ```
  mxhost1,mxhost2,mxhost3,...,mxhostN;rollupname
  ```

  or:

```
[mxhost-ip-1],[mxhost-ip-2],[mxhost-ip-3],[mxhost-ip-N];rollupname
```

Here "mxhost1...mxhostN" are the names of all the MX hosts included in the rollup, mxhost-ip-1...mxhost-ip-N are the IP addresses of all the MX hosts included in the rollup, and "rollupname" is the domain name associated with the rollup. In the case of a host name list, the list should be exactly the same as the list specified in the corresponding "rollupmx_" entry used by the `MX_ACCESS` mapping, and he domain name must end with the rollup domain suffix.

- If the initial lookup for a given host fails it is repeated with the first element of the host name replaced by a "*". If this fails a lookup is attempted with the first two elements replaced with a "*".
- If a match is found the corresponding entry value is checked to make sure all of the MXes it specified are also on the MX list for the destination domain. If they are a result is returned that adds the rollup name to the address as a source route. If not the process continues with the next host.

50.8.4.8.1 Setting up an MX rollup

A rollup consists of two sets of general database entries, one set to match the various MX hosts to the rollup and another to specify the MX hosts for the rollup. This sounds more complex than it actually is because the MX hosts in the first set have to match the set in the second.

For example, let's suppose that a large service provider, example.net, is known to provide service for many different domain, but all of the domains share the MX hosts mx1.example.net, mx2.example.net, and mx3.example.net. An appropriate set of database entries for this service provider would be:

```
rolluphost_mx1.example.net   mx1.example.net,mx2.example.net,mx3.example.net;example.net.rollup
rolluphost_mx2.example.net   mx1.example.net,mx2.example.net,mx3.example.net;example.net.rollup
rolluphost_mx3.example.net   mx1.example.net,mx2.example.net,mx3.example.net;example.net.rollup

rollupmx_example.net.rollup  mx1.example.net,mx2.example.net,mx3.example.net
```

If this seems highly duplicative, that's because it is: Denormalization is required to turn a bidirectional mapping into a set of entries in a name/value store.

In this case a wildcard could be used to lower the number of entries:

```
rolluphost_*.example.net     mx1.example.net,mx2.example.net,mx3.example.net;example.net.rollup

rollupmx_example.net.rollup  mx1.example.net,mx2.example.net,mx3.example.net
```

When wildcards are used it may be necessary to specify the matching list as a serious of IP addresses known to be associated with the service instead of a series of host names:

```
rolluphost_*.example.net     [ip1],[ip2],[ip3];example.net.rollup

rollupmx_example.net.rollup  mx1.example.net,mx2.example.net,mx3.example.net
```

50.8.4.9 mx_access callout

The `mx_access` callout is used in conjunction with the `forward` callout (described above) to implement a general purpose MX rollup facility.

Once the steps for setting up the `forward` callout have been done, the `mx_access` callout should be added to the `MX_ACCESS` mapping table as follows:

```
MX_ACCESS

   *|*.rollup     $C$[IMTA_LIB:smartsend.so,mx_access,<pflags>|$0.rollup]$E
```

The ".rollup" that appears on both sides of the mapping should be replaced with whatever domain is to be used to identify rollups.

The `mx_access` callout constructs a probe of the form `rollupmx_<domaint>`, where "domain" is the destination domain, and looks it up in the general database. If a match is found the value of the entry is expected to be of the form:

```
mxhost1,mxhost2,mxhost3,mxhostN
```

This value is then returned as the result, which then becomes the MX list for the domain.

50.8.4.10 mta_startup and mta_shutdown callouts

The `mta_startup` and `mta_shutdown` callouts are intended to be called from the `MTA_STARTUP` and `MTA_SHUTDOWN`mapping. These callouts should be set up as follows:

```
MTA_STARTUP

   *      $C$[IMTA_LIB:smartsend.so,mta_startup,<pflags>|$0]$E

MTA_SHUTDOWN

   *      $C$[IMTA_LIB:smartsend.so,mta_shutdown,<pflags>|$0]$E
```

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output. Only one additional bit is defined for both mappings: Bit 3, value 8, should be set as part of configuring open connection aggregation with the log_action callout described below.

Currently the callout always fails, so it can be safely inserted before any other actions in the mapping.

At present no other functionality is provided by these callouts, so they are not configured as part of the smartsend,rcp recipe.

50.8.4.11 log_action callout

The `log_action` callout is intended to be called from the `LOG_ACTION` mapping. It provides the ability to store counter information about MTA activity using a combination of Redis sets and sorted sets. This provides the ability to determine things like the most active domains, the domain generating the most temporary failures, and so on. Currently the callout always fails, so it can be safely inserted before any other actions in the mapping.

Note that this callout is only needed in specific circumstances and therefore is not configured by the smartsend.rcp recipe.

As part of setting up the callout, bits 1 and 4, values 2 and 16, respectively, of the log_conversion_tag MTA option should also be set if conversion tags are used so that the tags are included in the LOG_ACTION mapping probe. The LOG_ACTION mapping should be set up as follows:

```
LOG_ACTION

*      $C$[IMTA_LIB:smartsend.so,log_action,<pflags>|$0]
```

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output. The next three bits enable different sets of counters as described in the following sections.

50.8.4.11.1 Open SMTP Connection Aggregation (bit 3, value 8)

First, host + process pairs are tracked used the Redis set smtp_host_pid. (Note that this part of the counters is actually done by the mta_startup and mta_shutdown callouts decribed above.) Entries in this set have names of the form <omta-host>_<omta-pid>_<omta-type>. Here <omta-type> is the type of process, e.g. "smtp_client", "smtp_server", etc. All processes that engage in enqueue and dequeue operations will add entries for themselves when they initialize and remove them when they terminate.

Second, three per-process sorted sets are also maintained:

- smtp_domain_<omta-host>_<pid> <domain> <numeric-value>
- smtp_vmta_<omta-host>_<pid> <vmta> <numeric-value>
- smtp_ip_<omta-host>_<pid> <vmta>_<ip> <numeric-value>

The <numeric-value> is incremented each time a connection is opened and decremented each time one is closed.

Note that these sorted sets contain no useful information and have no pointer references when the associated MTA process terminate and are therefore deleted by the mta_shutdown callout.

Third, three global connection tracking sorted sets are maintained:

- smtp_domain_<genid> <domain> <numeric-value>
- smtp_vmta_<genid> <vmta> <numeric-value>
- smtp_ip_<genid> <vmta>_<ip> <numeric-value>

The <genid> is obtained from the Redis entry smtp_genid. If no such entry exists one is created with value "0".

Note that once entries in the global connection tracking sorted sets are created they will never be deleted; some external cleanup mechanism must be provided.

50.8.4.11.2 Queue Count Aggregations in Redis (bit 4, value 16)

First, hosts are tracked in the Redis set queue_host. Each entry is consists of a host name <host>. Note that no facility is provided to remove entries from this set. (Note that this addition is actually done by the mta_startup callout described above.)

Second, current queue message counts are tracked in a sorted set per host by two axes:

- `queue_domain_<host> <domain>_<channel> <numeric-value>`
- `queue_vmta_<host> <vmta>_<channel> <numeric-value>`

Here `<channel>` is the name of the MTA channel queue.

Third, two global queue counts are maintained:

- `queue_domain_<genid> <domain>_<channel> <numeric-value>`
- `queue_vmta_<genid> <vmta>_<channel> <numeric-value>`

The `<genid>` is obtained from the Redis entry `queue_genid`. If no such entry exists one is created with value "0".

Note that once entries in any of these sorted sets are created they will never be deleted; some external cleanup mechanism must be provided.

50.8.4.11.3 Success/Error Count Aggregations in Redis (bit 5, value 32)

These counters provide information about SMTP operation success/failure rates in a series of buckets, each represented by a sorted set:

- `<success|tempfail|permfail>_domain_<MM><00|30> <domain> <numeric-value>`
- `<success|tempfail|permfail>_vmta_<MM><00|30> <vmta> <numeric-value>`
- `<success|tempfail|permfail>_ip_<MM><00|30> <ip> <numeric-value>`

Here `<MM>` is the minute part of the current time. This has the effect of creating one bucker per minute.

Again, no mechanism exists for resetting the bucket contents, deleting entries from these buckets, or deleting the buckets themselves.

50.8.4.12 dequeue_access callout

The `dequeue_access` callout is intended to be called from the `DEQUEUE_ACCESS` mapping. It provides the ability to record overall message latency in the smartsend log field and to send this information in JSON format to a separate latency server for further processing.

Note that this callout is only needed in specific circumstances and therefore is not configured by the smartsend.rcp recipe.

As part of setting up the callout, bit 12, value 4096, of the `include_conversiontag` MTA option should also be set. (Latency computations depend on conversion tag information.) Bit 0 (value 1) of the `log_smartsend` MTA option should also be set in order for the latency information to be logged. The `DEQUEUE_ACCESS` mapping should then be set up as follows:

```
DEQUEUE_ACCESS

   *      $C$[IMTA_LIB:smartsend.so,dequeue_access,<pflags>|$0]
```

Here <pflags> should be replaced with an unsigned integer flag value. The lowest three bits enable increasing levels of debug output, with "0" resulting in no output.

Bit 3, value 8 of <pflags> enables writing of the calculated latency in the smartsend log field. Assuming that the time the message was received appears in the fifth conversion tag entry, expressed as an integer UTC time since the epoch in milliseconds (date +%s%3N on Linux), the elapsed time up to this delivery attempt will be computed and stored in the smartsend log field with the prefix "latency=".

Bit 4, value 16, enables writing of the calculated latency to the latency server.

50.8.4.13 MTA identity option: `id` (string)

The `id` MTA option specifies a name for a group of MTAs that share network connections. At present this name is only used by the smartsend facility in looking up IP address list information.

The official host name on the local channel is used if this option is not specified.

50.8.4.14 smartsend options: `smartsend_use_redis` (0 or 1)

The `smartsend_use_redis` smartsend option selects the backend database protocol used by the smartsend plugin. A value of 1 selects redis; a value of 0 selects memcache. The default if the option is not specified is to use redis if either the `redis.hostlist` or `redis.servicename` option is set, memcache otherwise.

50.8.4.15 Timeout for IP backoff entries (`ipbackofftimeout`)

An entry is made in memcache or Redis when IP backoff mode engages. This option sets the timeout value for the entry. The value is expressed in seconds, with a default value of 3600 (one hour).

Chapter 51 Message conversions

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 group, 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-CONVERSION` effects 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 converters. 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` entries (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.

The `imsimta test -translation` utility can perform character set conversions for testing or scripting purposes; the `imsimta test -mime` utility can perform certain message conversions, for testing purposes.

Note that certain basic choices regarding the fundamental handling of attachments/MIME parts can be configured for arbitrary channels via setting of channel options, potentially not requiring any of the more complex conversion channel or message reformatting facilities discussed here; for these basic choices, see the Attachments and MIME processing channel options.

51.1 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 converters 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 converters 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.

51.1.1 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 utility, 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 pattern (left hand side of the rule) or template (right hand side of the rule), thus often require quoting of pattern and/or template 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 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 cause an additional, compound field to be appended to the CONVERSIONS mapping table probe, immediately after any "TAG=" clause.

New in the 8.0.2.3 release, setting bit 0 (value 1) of the include_domain MTA option will casue an additiona field to be appended to the CONVERSIONS mapping table probe containing the destination domain for the current set of recipients, immediately after any "MTPRIORITY=" and "BLOCKS=" clauses.

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 (when a match on the pattern, *i.e.*, left hand side, occurred) using a $:T test in the template (right hand side) 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 51.1 Additional CONVERSIONS 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=*channel-name*" clause, is used to select some other sort of channel, such as a tcp_* 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=*channel-name*	Route through the alternate channel *channel-name* rather than the regular conversion channel
Delete	"Flatten" any message/rfc822 body part (forwarded message) into a message content part, deleting the forwarded headers
dkimidentity-N=identity	New in MS 8.0.2.3. Activate DKIM signing in slot N using using the specified identity. See the description of the `destinationdkimidentityN` channel option for additional information on the value's semantics. N can range from 0 to 3. This template keyword also clears the corresponding selector value, which can subsequently be set with the dkimselector-N template keyword.
dkimselector-N=selector	New in MS 8.0.2.3. Specifies the selector list for DKIM slot N. See the description of the `destinationdselectorN` channel option for additional information on the value's semantics. N can range from 0 to 3. This template keyword must be specified after the corresponding dkimidentity-N template keyword.
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 MS 6.3) Perform any configured charset conversion *before* routing to the conversion channel
QUOTED-PRINTABLE	Switch MIME encodings to QUOTED-PRINTABLE
Record,Text	Line wrap text/plain parts at 80 characters
Record,Text=*n*	Line wrap text/plain parts at *n* characters
RFC1154	Convert message to RFC 1154 format
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
```

51.1.1.1 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 remote users.)

Such "alternate conversion channel" routing has also found an important application in causing special routing of notification messages; see notification message routing.

51.1.1.2 Mapping table MTA options: `include_domain` (bitmask)

New in MS 8.0.2.3. This option selectively enables the inclusion of destination domain information in various mapping table probes. When enabled, the destination domain will be included in the probe in a table-specific way.

This option takes a bit-encoded integer value. The currently defined bits are shown in the following table:

Table 51.2 `include_domain` MTA option bit values

Bit	Value	Usage
0	1	CONVERSIONS : include in "DOMAIN=value" format after the block count

51.1.2 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.

51.1.3 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.*, *CONFIGROOT*/conversions. (In prior versions, the conversions file was located via the IMTA tailor file option imta_conversion_file, normally pointing to IMTA_TABLE: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 (see RFC 2045). 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"
```

51.1.3.1 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 specified IN-CHAN and OUT-CHAN parameters match the channels[1] through which the message is passing;
- and if the specified PART-NUMBER matches the structured part number[2] of the message part;
- and if the (required!) IN-TYPE parameter, as well as all specified IN-PARAMETER-NAME, IN-PARAMETER-VALUE, and IN-SUBTYPE parameters, match the Content-type: of the message part;
- and if all specified 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 specified IN-SUBJECT, IN-MESSAGE-CONTEXT, IN-A1-TYPE, and IN-A1-FORMAT values match those of the headers of the immediately enclosing message (message/rfc822 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).

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.*.

51.1.3.2 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 51.3 Available conversion parameters, grouped by functionality

Parameter	Meaning
Command parameters	
CALL	Routine to call. The argument takes the form *image|routine|argument*.
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 *image|routine|argument*. 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, ...
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, ...
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, *e.g.*, *a.b.c*... 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`, ... 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`, ... 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`, ... 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-`*n*	Output MIME Content-disposition parameter name; *n* = `0`, `1`, `2`, ...
`OUT-DPARAMETER-VALUE-`*n*	Output MIME Content-disposition parameter value corresponding to `OUT-DPARAMETER-NAME-`*n*.
`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-*n*	Output MIME Content-type parameter name; *n* = 0, 1, 2, ...
OUT-PARAMETER-VALUE-*n*	Output MIME Content-type parameter value corresponding to OUT-PARAMETER-NAME-*n*.
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-*n*	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; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to copy, as matched by an IN-PARAMETER-NAME-*m* 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.

51.1.3.3 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.

51.1.3.4 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. (In constrast, Environment variables for use by conversion channel shell procedures below shows environment variables available for use *within a conversion shell script procedure*.)

Table 51.4 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 51.5 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.
content-disposition parameters	
content-type parameters	

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 51.6 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.

51.1.3.5 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"
```

51.1.3.6 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:.

51.1.3.7 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 51.7 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)

51.2 Conversion tags

The MTA has a private mechanism of *conversion tags*. Conversion tags 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 documention 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 messages 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 the include_conversiontag MTA option), 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.

51.3 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.

51.3.1 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.

As of MS 8.0.2.1, enabling charset conversions also checks for and attempts to mitigate the various so-called MailSploit attacks. More specifically, if charset conversions are enabled for a given destination, the MTA will:

1. Check the content of and remove unnecessary encoded-words during submission. In particular, encoded-words consisting of nothing but an atom or domain will be decoded. Note that this happens only when using SUBMIT, which should always be before DKIM or similar protection mechanisms are applied.

2. Remove control characters other than those needed for MIME-compatible charsets. This includes, but is not limited to NUL, CR and LF.

3. Remove any empty encoded-words that result from (2).

4. Recognize and process encoded-words outside the contexts where they normally appear. (This behavior is appropriate for an MTA seeking to mitigate attacks that may depend on broken clients that recognize encoded-words outside the contexts where they normally occur.)

Note that the removal of NULs and similar CTLs is justified from a standards perspective since RFC 2047 requires that encoded words represent printable material in a MIME text/US-ASCII compatible charset. As such, this material should not be present in standards-compliant encoded-words.

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 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 51.8 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=`*n*	Line wrap text/plain parts at *n* 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 pattern, *i.e.*, left hand side, occurred) using a `$:T` test in the template (right hand side) output string. Such tests are more commonly used in the `CONVERSIONS` mapping table, but under less common conditions may potentially be useful here in the `CHARSET-CONVERSION` mapping table.

51.3.1.1 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-UUDECODE,
- DEFLATE-UUENCODE,
- HEXADECIMAL,
- OLD-BASE64,
- PATHWORDS,
- QUOTED-PRINTABLE,
- UUCODE,
- UUDECODE,
- UUENCODE,
- X-ATOB,
- X-BASE32,
- X-BASE85,
- X-BINHEX,
- X-BTOA,
- X-C-DATA,
- X-COMPRESSED-UUCODE,
- X-COMPRESSED-UUDECODE,
- X-COMPRESSED-UUENCODE,
- X-DEFLATE-UUCODE,
- X-DEFLATE-UUDECODE,
- X-DEFLATE-UUENCODE,

- X-HEXADECIMAL,
- X-OLD-BASE64,
- X-PATHWORKS,
- X-UUCODE,
- X-UUDECODE,
- 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.

51.3.1.1.1 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
```

51.3.2 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.

51.3.2.1 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 undiscriminated 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
```

51.3.2.2 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.

51.3.2.2.1 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.

51.3.2.2.1.1 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
```

```
   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
```

51.3.3 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 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. (See Conversion control for additional details on conversions entries.)

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.)

51.3.4 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. 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 51.9 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 `IMTA_TABLE:conversions`. As of Messaging Server 7.0, the conversions file is located at `config-root/conversions`.

51.4 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 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 52 MTA options

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 specificially 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.

52.1 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 editted 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 editting 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.

52.2 Legacy configuration MTA option file

The MTA legacy configuration option file is the file `CONFIGROOT/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 tailing 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 shart 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.

52.2.1 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 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.)

52.3 Getting option changes to take effect on the MTA

Options, both MTA options and channel options, are part of the core 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" core MTA configuration. When a compiled configuration is in use (exists), then the current "live" core 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 most MTA processes do automatically "cycle" --- that is, time out and expire to be replaced with new processes --- changes to the "live" configuration generally will get seen eventually, if not immediately, even if the `reload` utility is not used. (In particular, MTA server processes, such as SMTP server processes, and channel delivery job processes, do expire and get replaced regularly, if not immediately; such channel processes are usually the processes that need to "see" changes to configuration options. However, the two main control processes for the MTA, namely the Dispatcher and Job Controller, run indefinitely and do not automatically expire, so for option changes affecting them, waiting will not suffice and a `reload` will be required.)

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).

52.4 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 52.1 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.e.*, 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
`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 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
alternate_recipient	(New in MS 8.0.1) Specify the comment string used to add an alternate recipient for an address.
alternate_recipient_mode	(New in MS 8.0.1) Control handling of multiple alternate recipients.
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 MS 6.2) Size of the cache of load average values
averages_cache_timeout	(New in 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
channel_table_size	Set the number of channels allowed in the MTA configuration
circuitcheck_completed_bins	Specify the bins for the message circuit check message counters

circuitcheck_paths_size	Number of circuit check paths (entries) to allow in the circuit check configuration file
comment_chars	Set the "comment" character(s) in MTA configuration files
content_return_block_limit	Force NOTARY non-return of content flag for messages over the specified size
conversion_size	Set the number of entries allowed in the conversion file (legacy configuration) or conversions MTA option (Unified Configuration)
conversions	(Unified Configuration only) Conversion entries; replaces legacy configuration conversions file
defer_group_processing	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
delimiter_char	(New in 7.0) RESTRICTED. Specify the delimiter character (normally the vertical bar character)
delivery_options	Specify interpretation of mailDeliveryOption values
dequeue_debug	Enable debugging of message dequeue operations
dequeue_map	RESTRICTED. Map files into memory when dequeuing
describe_cache_limit	Control size of message part description cache for message body processing purposes
digest_on	Specify the comment string that enables mailing list digests (in preference to regular mailing list postings); usage is **RESTRICTED**
dirsync_hack	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**
domain_failure	Rewrite template to apply when an LDAP lookup encounters an LDAP error
domain_hash_size	Set the number of rewrite rules allowed
domain_match_cache_size	Number of domain lookup ($V rewrite template) results to keep cached
domain_match_cache_timeout	How long to cache domain lookup ($V rewrite template) results
domain_match_url	URL for an extra, special lookup for domains, *e.g.*, 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
duplicate_tracking_url	(New in 8.0) Template for location of per-user duplicate message tracking files
enable	Enable operation of the MTA

enable_sieve_regex	Control whether Sieve filters may use the regex extension (the :regex match-type)
error_text_*	Specify alternate error text for any of various error conditions; see Table of error text options
expandable_default	Set the default for mailing lists to [NONEXPANDABLE], *i.e.*, disable use of SMTP EXPN
exproute_forward	Control whether the exproute channel option affects forward pointing headers
file_member_size	Maximum number of configuration files used by the MTA
filter_cache_size	New in 6.2. Specify the size of the per-process cache of tokenized Sieve filters
filter_cache_timeout	New in 6.2. Specify the retention time for entries in the per-process cache of tokenized Sieve filters
filter_debug	New in 6.2. Control whether detailed (stack state) debugging of Sieve evaluation is output.
filter_discard	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 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
forward_data_size	Hash size for the forward (database) text file
fruits_size	Number of fruits allowed in the fruit validation table
fsync	Do an fsync upon file close
general_case	Controls the case sensitivity of general database lookups
general_data_size	Hash size for the general (database) text file
group_dn_template	Template used when looking up a uniqueMember of a group
header_limit	Sets a maximum size for the primary (outermost) message header
held_sndopr	Send operator or syslog messages when messages are HELD
history_to_return	Control the amount of delivery attempt history included in bounced messages
host_hash_size	Set the number of channel host names
id_domain	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. Include transport and application connection information in mapping table probes
include_conversiontag	New in 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 service.defaultdomain 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 Direct LDAP attribute name MTA 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 *vs*. fetches only "known" domain attributes
ldap_domain_root	The base DN for the domain portion of the DIT; overrides the service.dcroot configutil parameter (legacy configuration) or dcroot base option (Unified Configuration)
ldap_domain_timeout	New in 6.1-0.01. Specify the retention time for entries in the domain map cache
ldap_global_config_templates	Schema 2 support: specify the DN where global configuration templates can be found
ldap_group_object_classes	The object classes required for a group

ldap_host	Host to which to connect for LDAP queries; overrides the local.ugldaphost configutil parameter (legacy configuration) or ugldaphost base option (Unified Configuration)
ldap_host_alias_list	Local host aliases; overrides (for MTA purposes) the local.imta.hostnamealiases 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 "l" channel); overrides (for MTA purposes) the local.hostname configutil parameter (legacy configuration) or hostname base option (Unified Configuration)
ldap_mail_aliases	The attributes in which aliases are stored; overrides the local.imta.mailaliases configutil parameter
ldap_mail_reverses	The attributes used for the filter generated by the $Q LDAP substitution sequence; normally used during reverse_url lookups, hence normally the attributes compared against an address to be "reversed"
ldap_max_connections	The maximum number of simultaneous LDAP connections to allow
ldap_object_class_hash_size	Internal hash size for the list of object classes relevant to the MTA
ldap_pab_host	New in 7.0. Usage: RESTRICTED. The host for PAB LDAP queries; overrides (for MTA purposes) the local.service.pab.ldaphost configutil parameter (legacy configuration) or ldaphost PAB option (Unified Configuration)
ldap_pab_password	New in 7.0. Usage: RESTRICTED. The password for PAB LDAP queries; overrides (for MTA purposes) the local.service.pab.ldappasswd configutil parameter (legacy configuration) or ldappasswd PAB option (Unified Configuration)
ldap_pab_port	New in 7.0. Usage: RESTRICTED. The port for PAB LDAP queries; overrides (for MTA purposes) the local.service.pab.ldapport configutil parameter (legacy configuration) or ldapport PAB option (Unified Configuration)
ldap_pab_username	New in 7.0. Usage: RESTRICTED. The username (bind credentials) for PAB LDAP queries; overrides (for MTA purposes) the local.service.pab.ldapbinddn 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

	`local.ugldapbindcred` 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 `local.ugldapport` 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 `local.imta.schematag` configutil parameter
`ldap_timeout`	Timeout value for LDAP queries
`ldap_uid_invalid_chars`	Characters that are not allowed in a `uid` or `permanent identifier`
`ldap_use_async`	Control use of asynchronous (*vs*. 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 `local.ugldapbasedn` 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 `local.ugldapbinddn` 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 transaction log entries
`log_connection`	Generate connection log entries, and/or include connection information in message transaction log entries. (New in 6.2, optionally log SMTP AUTH attempts.)
`log_connections_syslog`	Send connection log entries to syslog (UNIX) or event log (NT)

log_conversion_tag	(New in 7.0.5.) Include conversion tag(s) field in MTA message transaction log entries.
log_delay_bins	Specify the bins for delivery delay range counters
log_delivery_flags	(New in 7.0.5.) Include delivery flags field in MTA message transaction log entries.
log_diagnostics	(New in MS 7.0u1) Include diagnostics in MTA message transaction log entries
log_dkim	(New in 8.1.0.6) Include information about DKIM signing operations in message transaction log entries
log_envelope_id	New in 6.1-0.01. Include envelope ids in MTA message transaction log entries.
log_filename	Include message file names in MTA message transaction log entries
log_filter	Include Sieve filter actions applying to the message in MTA message transaction log entries.
log_format	Control the format of the MTA transaction log file(s)
log_from	(New in 8.1.0.2) Include the address found in the From: header field in message transaction log entries
log_futurerelease	(New in) Include value of FUTURERELEASE SMTP extension in MTA message transaction log entries
log_header	Include message headers in MTA message transaction log entries
log_imap_flags	(New in 7.0.5.) Include any IMAP flags set by the MTA in MTA message transaction log entries
log_intermediate	New in 6.2. Include "intermediate" and/or "original" (RCPT TO: command line) forms of destination address in MTA message transaction log entries
log_isc_status	(New in MS 8.0.2) Include Indexed Search Converter status information in LMTP server MTA message transaction log entries
log_local	Include the local domain name on "bare user name" addresses in MTA message transaction log entries
log_mailbox_uid	Include the IMAP UID and UIDVALIDITY of messages delivered by the ims-ms channel to the Message Store in MTA message transaction log entries
log_message_id	Include message IDs in MTA message transaction log entries
log_messages_syslog	Send MTA message transaction log file entries to syslog

log_mtpriority	(New in MS 8.0) Include message MT-PRIORITY in MTA message transaction log entries
log_node	OpenVMS only. Include the node name for an enqueueing process in MTA message transaction log entries
log_notary	Include a NOTARY (delivery receipt) flags indicator in MTA message transaction log entries
log_priority	Include message priority in MTA message transaction log entries
log_process	Include enqueuing process ID in MTA message transation log entries
log_queue_time	(New in 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
log_reason	Include the reason for a message rejection in MTA message transaction log entries
log_remota_mta	(New in MS 8.0.2.3) Include remote MTA name in MTA message transaction log entries
log_sensitivity	Include message's sensitivity value in MTA message transaction log entries
log_size_bins	Specify the bins for message size range counters
log_smartsend	(New in 8.1.0.1) Include additional information about smartsend plugin actions in MTA message transaction log entries
log_sndopr	Send a syslog message if the MTA's logging facilities encounter a difficulty
log_syslog_prefix	(New in MS 8.0.2.3.) Specifices the prefix used on MTA message transaction log file entries sent to syslog
log_times	(New in 8.0) Include deferred delivery time in MTA message transaction log entries
log_tracking	(New in 8.0) Include message tracking ID in MTA message transaction log entries
log_transactionlog	(New in 8.0) Include string argument from any Sieve "transactionlog" actions in MTA message transaction log entries
log_uid	(New in 8.0) Include recipient UID in MTA message transaction log entries
log_use_xtext	(New in MS 8.0) Controls xtext encoding of addresses in MTA message transaction log entries
log_username	Include user identity information in MTA message transation log entries
log_8bit_encode	Controls how non-ASCII characters are written in XMl format MTA transaction log file(s)

`logfile`	`logfile` options set at the MTA level (`mta.logfile.`*`option-name`* in Unified Configuration) affect the MTA's logging of insertion of messages into the Message Store; that is, affect logging of message insertions performed by `ims-ms` and `tcp_lmtpss_*` channels
`mail_off`	Specify the comment string that disables mail delivery for list addresses
`map_names_size`	Set the number of mapping tables
`mapping_paranoia`	(New in 7.0) Control handling of vertical bar characters in probes of mapping tables such as FROM_ACCESS, recipient access mapping tables and `AUTH_REWRITE`
`max_addheaders`	Maximum number of Sieve "`addheader`" actions that can be performed
`max_alias_levels`	Set the level of alias nesting allowed
`max_duplicates`	(New in 8.0) Maximum "`duplicate`" tests performed in a Sieve filter
`max_fileintos`	Maximum number of Sieve "`fileinto`" actions that may be performed by a Sieve script
`max_header_block_user`	Fine tune message fragmentation
`max_header_line_user`	Fine tune message fragmentation
`max_internal_blocks`	Specify size of messages beyond which to buffer to temporary files
`max_local_received_lines`	Occurrences of the local host name in Received: header lines after which a message will be HELD
`max_mime_levels`	Degree to look inside MIME messages during processing
`max_mime_parts`	Number of parts to look at when processing MIME messages
`max_mr_received_lines`	Number of MR-Received: header lines after which a message will be HELD
`max_received_lines`	Number of Received: header lines after which a message will be HELD
`max_redirects`	Maximum number of Sieve "`redirect`" actions (*i.e.*, forwards) that may be used in a Sieve filter
`max_total_received_lines`	Number of Received:, MR-Received: or X400-Received: header lines after which a message will be HELD
`max_urls`	Maximum number of URLs that may be active (nesting of references)
`max_vacations`	Maximum number of `vacation` actions that may appear in a Sieve filter
`max_variables`	Maximum number of variables that may be used in a Sieve script

max_x400_received_lines	Number of X400-Received: header lines after which a message will be HELD
memcache_expire	(New in 8.0) Time to hold idle memcached connections open
memcache_hash_algorithm	(New in 8.1.0.3) Hash function to apply to memcache keys
memcache_host	(New in 8.0) Host name of memcached server
memcache_port	(New in 8.0) memcached service port
message_hash_algorithm	(New in MS 6.3-0.15) Algorithm to use for generating message hashes for message archiving
message_hash_fields	(New in MS 6.3) Header fields to include when generating message hashes
message_save_copy_flags	(New in MS 6.3) Control the format of MESSAGE-SAVE-COPY mapping table probes
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
normal_block_limit	Maximum size of message to treat as being of normal or higher priority
non_urgent_block_limit	Maximum size of message to treat as being of non-urgent priority
notary_decode	Control whether encoded-words in the original message header are decoded when performing a %H substitution during the MTA's generation of DSNs and MDNs
notary_quote	Specify the character that marks substitution sequences in return_*.txt files and disposition_*.txt files
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
optin_user_carryover	New in 6.2. Specify whether user spam/virus filter "opt-in" is considered to "carry over" for forwarded recipients
options_hash_size	Hash size for the internal table of MTA options (option.dat options)

or_clauses	Set default for whether to AND or OR multiple posting access control settings on mailing lists
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
post_off	Specify the comment string that disables posting for list addresses
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
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_body_needed	Specifies whether or not to retain decoded MIME body information when initial message analysis is performed.

sieve_mime_needed	Specifies whether or not to retain inner MIME structure and header information when initial message analysis is performed.
sieve_redirect_add_resent	(New in 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_prefix	(New in MS 8.0.2.3.) Set the prefix attached to syslog notices.
sndopr_priority	Set the syslog level (facility and priority) the MTA uses when generating syslog notices
spamfilter*N*_action_*M*	URL that resolves to a Sieve filter, specifying the action to take when Spam/virus filter package *N* returns the corresponding spamfilter*N*_verdict_*M* verdict
spamfilter*N*_config_file	Location of the Spam/virus filter package configuration file
spamfilter*N*_final	Control whether the MTA passes the "intermediate" *vs*. "final" address form to the spam/virus filter package
spamfilter*N*_library	Location of the Spam/virus filter package client shared library
spamfilter*N*_name	(New in MS 7.0.5) Specify a symbolic name for the *N*th spam/virus filter package.
spamfilter*N*_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
spamfilter*N*_null_optin	New in 6.1-0.01. Specify a value for the attribute named by the ldap_domain_attr_optin*N* MTA options, the ldap_optin*N* MTA options, or the (new in MS 8.0.1.3) ldap_optout*N* MTA options that means that the attribute should be ignored; the purpose is to provide a way for these attributes to be used by provisioning facilities that find it easier to set an attribute value rather than to remove an attribute entirely
spamfilter*N*_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
spamfilter*N*_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
string_pool_size_1	Set the number of strings allowed for MTA mapping table use
string_pool_size_2	Set the number of strings allowed for MTA alias use
string_pool_size_3	Set the number of strings allowed for MTA general (database) text file use
string_pool_size_4	Set the number of strings allowed for MTA personal conversion use
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 addresses for SRS purposes
tracking_debug	(New in 8.0) Enable debugging of message tracking
tracking_hash_algorithm	(New in 8.1.0.3) Select the hash algorithm the MTA uses to generate hashes of tracking and recall secrets. The default is SHA-1.
tracking_mode	(New in 8.0) Enable message tracking data storage
tracking_retries	(New in 8.0) Number of tracking update retry attempts
tracking_retry_delay	(New in 8.0) Time delay between tracking update retry attempts
unique_id_template	(New in MS 6.3) Specify a template used to convert an address into a unique identifier; typically used in conjunction with archiving facilities
urgent_block_limit	Maximum size of message to treat as being of urgent priority
use_alias_database	Control use of the alias database
use_comment_strings	Control the format of COMMENT_STRINGS mapping table probes
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_personal_names	Control the format of PERSONAL_NAMES mapping table probes
use_permanent_error	Control whether certain error conditions are considered to be permanent errors, or temporary errors
use_reverse_database	Control use and format of the reverse mapping table and reverse database

use_temporary_error	(New in 8.0.1) Control whether certain error conditions are considered to be temporary errors, or permanent errors
use_text_databases	Control whether the general database, the reverse database, and the forward database are on-disk databases, or in-memory structures
url_result_cache_case	New in MS 8.1.0.2. Controls whether URLs are treated in a case-sensitive fashion.
url_result_cache_size	Number of rewrite rule and mapping table LDAP URL lookups ($]...[lookups) to keep cached
url_result_cache_timeout	How long to cache rewrite rule and mapping table LDAP URL lookups
user_case	Lower case postmaster; other usernames with localbehavior channel option
vacation_cleanup	Average number of times between cleanups of the per-user per-response vacation files
vacation_cleanup	(New in MS 8.1.0.3) Specifies the hash algorithm to use for creating vacation entry names
vacation_maximum_timeout	(New in MS 7.0.5) Specify a maximum value permitted for Sieve "vacation" periodicity arguments
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
wild_pool_size	Set the total number of wildcards allowed in mappings' patterns

52.5 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 52.2 MTA options, listed by functionality

Option	Usage
enable	Enable operation of the MTA
Alias and address MTA options	
alias_case	RESTRICTED: Control the case sensitivity of aliases

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
alternate_recipient	(New in MS 8.0.1) Specify the comment string used to add an alternate recipient for an address.
alternate_recipient_mode	(New in MS 8.0.1) Control handling of multiple alternate recipients.
ap_debug	Internal MTA debugging option: enable debugging of low level address parsing; support for this option is RESTRICTED
delimiter_char	(New in MS 7.0) RESTRICTED: ASCII position of delimiter character
exproute_forward	Control whether the exproute channel option affects forward pointing headers
idn_config_file	(New in MS 8.0.2) Specify the location of an optional IDNKIT configuration file
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_cleanup	(New in MS 8.1.0.3) Specifies the hash algorithm to use for creating vacation entry names
vacation_maximum_timeout	(New in MS 7.0.5) Specify a maximum value permitted for Sieve " vacation" periodicity arguments
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 service.defaultdomain configutil parameter)
ldap_domain_root	The base DN for the domain portion of the DIT; overrides the dcroot base option (legacy configuration) service.dcroot configutil parameter)
ldap_host	Host to which to connect for LDAP queries; overrides the ugldaphost base option (legacy configuration local.ugldaphost configutil parameter)

ldap_host_alias_list	Local host aliases; overrides (for MTA purposes) the ldap_host_alias_list base option (legacy configuration local.imta.hostnamealiases configutil parameter)
ldap_local_host	The local host name (official host name for the "l" channel); overrides (for MTA purposes) the hostname base option (legacy configuration local.hostname configutil parameter)
ldap_mail_aliases	The attributes in which aliases are stored; overrides the legacy configuration local.imta.mailaliases 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 local.service.pab.ldaphost 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 local.service.pab.ldappasswd configutil parameter)
ldap_pab_port	(New in MS 7.0) The port for PAB LDAP queries; overrides the ldapport PAB option (legacy configuration local.service.pab.ldapport 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 local.service.pab.ldapbinddn configutil parameter)
ldap_password	The password to use when binding for LDAP queries; overrides (for MTA purposes) the ugldapbindcred base option (legacy configuration local.ugldapbindcred configutil parameter)
ldap_port	Port to which to connect for LDAP queries; overrides (for MTA purposes) the ugldapport base option (legacy configuration local.ugldapport configutil parameter)
ldap_schematag	The tag (name) of the schema in use; overrides (for MTA purposes) the legacy configuration local.imta.schematag configutil parameter
ldap_username	The DN under which to bind for LDAP queries; overrides (for MTA purposes) the

	uglдapbinddn base option (legacy configuration local.ugldapbinddn 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 local.ugldapbasedn 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)
conversions	(Unified Configuration only) Conversion entries; replaces legacy configuration conversions file
include_conversiontag	(New in 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
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 *vs.* "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_case	Controls the case sensitivity of general database lookups
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
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, *e.g.*, file operations
return_debug	Enable debugging of MTA periodic return job operations
return_verify	(New in MS 7.0.5, replacing former imta_return_verify MTA Tailor option) Enable shell script logging within the MTA return job
tracking_debug	RESTRICTED: (New in MS 8.0) Enable debugging of the message 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 local.ugldaphost configutil parameter (legacy configuration) or ugldaphost base option (Unified Configuration)
ldap_max_connections	The maximum number of simultaneous LDAP connections to allow
ldap_password	The password to use when binding for LDAP queries; overrides (for MTA purposes) the local.ugldapbindcred 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 local.ugldapport configutil parameter (legacy configuration) or ugldapport base option (Unified Configuration)
ldap_timeout	Timeout value for LDAP queries
ldap_use_async	Control use of asynchronous (*vs.* synchronous) LDAP lookups
ldap_username	The DN under which to bind for LDAP queries; overrides (for MTA purposes) the local.ugldapbinddn configutil parameter (legacy configuration) or ugldapbinddn 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 ($V rewrite template) results to keep cached

domain_match_cache_timeout	How long to cache domain lookup ($V rewrite template) results
domain_match_url	URL for an extra, special lookup for domains, *e.g.*, 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 ldap_global_config_templates MTA option) that is used to store the domain search filter template
ldap_basedn_filter_schema1	(New in 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 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 defaultdomain base option (legacy configuration service.defaultdomain configutil parameter)
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 *vs.* 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) service.dcroot configutil parameter)
ldap_domain_timeout	New in MS 6.1-0.01. Specify the retention time for entries in the domain map cache
ldap_host_alias_list	Local host aliases; overrides (for MTA purposes) the local.imta.hostnamealiases 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 "l" channel); overrides (for MTA purposes) the local.hostname configutil parameter (legacy configuration) or hostname base option (Unified Configuration)

Direct LDAP usergroup lookup MTA options	
alias_entry_cache_negative	Whether to cache negative results (*i.e.*, 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_url0	URL for doing alias lookups
alias_url1	URL for doing alias lookups
alias_url2	URL for doing alias lookups
alias_url3	URL for doing alias lookups
allow_unquoted_addrs_violate_rfc2798	Look up addresses in LDAP with quotes stripped from the local part
aliasdetourhost_null_optin	(New in 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
defer_group_processing	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel
group_dn_template	LDAP URL template used for fetching attributes once a user specified via a uniqueMember attribute has been located
ldap_default_attr	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
ldap_filter_reference	(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 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 mailSieveRuleSource. 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 mail.)

ldap_group_object_classes	The object classes required for a group
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_mail_aliases	The attributes in which aliases are stored; overrides the legacy configuration local.imta.mailaliases configutil parameter
ldap_mail_reverses	The attributes used for the filter generated by the $Q LDAP substitution sequence; normally used during reverse_url lookups, hence normally the attributes compared against an address to be "reversed"
ldap_parental_controls	(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 Yes, 1, or true is considered to be requesting parental controls.
ldap_uid_invalid_chars	Characters that are not allowed in a uid or permanent identifier
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 ugldapbasedn base option (legacy configuration local.ugldapbasedn configutil 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 inetDomainSearchFilter.
ldap_basedn_filter_schema1	(New in 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 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 *vs*. 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) service.dcroot configutil 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 local.imta.schematag configutil parameter
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 ugldapbasedn base option (legacy configuration local.ugldapbasedn configutil parameter)
Direct LDAP attribute interpretation MTA options	
aliasdetourhost_null_optin	(New in 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 service.defaultdomain 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 local.imta.hostnamealiases 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 "l" channel); overrides (for MTA purposes) the local.hostname configutil parameter (legacy configuration) or hostname base option (Unified Configuration)
ldap_uid_invalid_chars	Characters that are not allowed in a uid or permanent identifier
optin_user_carryover	(New in MS 6.2.) Specify whether user spam/virus filter "optin" is considered to "carry over" for forwarded recipients
process_substitutions	(New in MS 6.3) Process substitution characters in the URL values of various LDAP attributes
route_to_routing_host	Forcibly route non-local hosts to the first value of mailRoutingHosts
sieve_user_carryover	Specify whether user Sieve filters "carry over" for forwarded recipients
spare_N_separator	(New in MS 7.0u2) Specify the handling of multiple values of the ldap_spare_N attribute for corresponding N.
vacation_maximum_timeout	(New in MS 7.0.5) Specify a maximum value permitted for Sieve "vacation" periodicity arguments
vacation_minimum_timeout	(New in 7.0.5) Specify a minimum value permitted for Sieve "vacation" periodicity arguments
Direct LDAP attribute name MTA options	
ldap_*	Specify an alternate attribute name for any of various per-user attributes; see MTA LDAP attribute name options
ldap_attr_domain_*	Specify an alternate attribute name for any of a few fundamental defining attributes of domains; see MTA LDAP attribute name options

ldap_domain_attr_*	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	
ldap_auth_attr_hold_for	(New in MS 8.0) Specify the name of an attribute that stores authenticated hold time
ldap_auth_attr_mail_host	(New in MS 8.0) Specify an alternate attribute name for the attribute used to store the mailHost for BURL purposes
ldap_auth_attr_recall_secret	(New in MS 8.0) Specify the name of an attribute used to store the user's recall secret
ldap_auth_attr_sender	(New in MS 8.0) Specify an alternate attribute name for the attribute used to store the user's canonical address
ldap_auth_attr_submit_channel	(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	
alias_entry_cache_negative	Whether to cache negative results (*i.e.,* 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
cache_debug	Output direct LDAP lookup caching statistics
domain_match_cache_size	Number of domain lookup ($V rewrite template) results to keep cached
domain_match_cache_timeout	How long to cache domain lookup ($V rewrite template) results
filter_cache_size	(New in MS 6.2.) Specify the size of the per-process cache of tokenized Sieve filters
filter_cache_timeout	(New in MS 6.2.) Specify the retention time for entries in the per-process cache of tokenized Sieve filters
ldap_domain_timeout	New in MS 6.1-0.01. Specify the retention time for entries in the domain map cache
ldap_timeout	Timeout value for LDAP queries
ldap_use_async	Control use of asynchronous (*vs.* synchronous) LDAP lookups
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
url_result_cache_case	New in MS 8.1.0.2. Controls whether URLs are treated in a case-sensitive fashion.

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)
Directory location MTA options	
langdir	(New in MS 7.0) Specify the location of the default set of language-specific files
tmpdir	(New in MS 7.0) Specify the location of the directory for storing temporary files
DKIM MTA options	
dkim_ignore_domains	(New in MS 7.0.5) Domains to ignore in DKIM-Signature: fields
dkim_preserve_domains	(New in MS 7.0.5) Domains in DKIM-Signature: fields that trigger passthrough mode
dkim_remove_domains	(New in MS 7.0.5) Domains in DKIM-Signature: fields triggering removal of the header
DNS lookup MTA options	
blocked_mail_from_ips	(New in 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 *vs.* 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.
`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, *e.g.*, 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
conversion_size	Set the number of entries allowed in the conversion file (legacy configuration) or conversions MTA option (Unified Configuration)
describe_cache_limit	Control size of message part description cache for message body processing purposes
domain_hash_size	Set the number of rewrite rules allowed
file_member_size	Maximum number of configuration files used by the MTA
forward_data_size	Hash size for the forward (database) text file
fruits_size	Number of fruits allowed in the fruit validation table
general_data_size	Hash size for the general (database) text file
host_hash_size	Set the number of channel host names
ldap_attr_name_hash_size	Internal hash size for the list of LDAP attributes relevant to the MTA
ldap_object_class_hash_size	Internal hash size for the list of object classes relevant to the MTA
map_names_size	Set the number of mapping tables
options_hash_size	Hash size for the internal table of MTA options (option.dat options in legacy configuration; mta.*option-name* options in Unified Configuration)
personal_conversion_size	RESTRICTED: Maximum personal conversion entries

reverse_data_size	Internal hash size for the reverse (database) text file
string_pool_size_0	Set the number of strings allowed for miscellaneous MTA configuration use
string_pool_size_1	Set the number of strings allowed for MTA mapping table use
string_pool_size_2	Set the number of strings allowed for MTA alias use
string_pool_size_3	Set the number of strings allowed for MTA general (database) text file use
string_pool_size_4	Set the number of strings allowed for MTA personal conversion use
wild_pool_size	Set the total number of wildcards allowed in mapping table patterns
LDAP external directory lookup MTA options	
ldap_ext_host	(New in MS 7.0u2) The host for LDAP EXT queries
ldap_ext_max_connections	(New in MS 7.0u2) Limit on the maximum number of LDAP EXT connections
ldap_ext_password	(New in MS 7.0u2) The password for LDAP EXT queries
ldap_ext_port	(New in MS 7.0u2) The port for LDAP EXT queries
ldap_ext_username	(New in MS 7.0u2) The username (bind credentials) for LDAP EXT queries
LDAP PAB lookup MTA options	
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 local.service.pab.ldaphost 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 local.service.pab.ldappasswd configutil parameter (legacy configuration)
ldap_pab_port	(New in MS 7.0) The port for PAB LDAP queries; overrides the local.service.pab.ldapport 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 local.service.pab.ldapbinddn configutil parameter (legacy configuration)
Mailing list and group MTA options	

alternate_recipient	(New in MS 8.0.1) Specify the comment string used to add an alternate recipient for an address.
alternate_recipient_mode	(New in MS 8.0.1) Control handling of multiple alternate recipients.
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 expandable: none or equivalently mgmanMemberVisibility: none; *i.e.*, 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 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 MS 6.3) Include the conversion tag value(s) in mapping table probes
include_domain	(New in MS 8.0.2.3) Include the destination domain in various 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_spares1	(New in MS 7u2, renamed from include_spares in 8.0.2.2) 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 various mapping table probes, especially *_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 MS 6.3) Include the conversion tag value(s) in mapping table probes
include_domain	(New in MS 8.0.2.3) Include the destination domain in various 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_spares2	(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
message_save_copy_flags	(New in 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_case	New in MS 8.1.0.2. Controls whether URLs are treated in a case-sensitive fashion.

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
enable_sieve_redis	(New in MS 8.0.2.3) Control whether Sieve filters may use the private redis extension
forward_database_url	(New in MS 8.0) memcache: URL for storing forward database data
general_case	Controls the case sensitivity of general database lookups
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
memcache_expire	(New in MS 8.0) Time to hold idle memcached connections open
memcache_hash_algorithm	(New in 8.1.0.3) Hash function to apply to memcache keys
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 *vs*. 2007 "journal" style
message_hash_algorithm	(New in MS 6.3) Algorithm to use for generating message hashes for message archiving
message_hash_fields	(New in MS 6.3) Header fields to include when generating a message hash
unique_id_template	(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	
block_limit	Limit the size of messages allowed through the MTA
block_size	Set the size of MTA "blocks"

bounce_block_limit	Limit the amount of original message content included in bounce messages
content_return_block_limit	Force NOTARY non-return of content flag for messages over the specified size
error_text_block_over	Specify error text returned in some cases of exceeding a destination channel blocklimit
error_text_line_over	Specify error text returned in some cases of exceeding a destination channel linelimit
error_text_list_block_over	Specify error text returned in some cases of exceeding a list's [BLOCKLIMIT] or mgrpMsgMaxSize value
error_text_list_line_over	Specify error text returned in some cases of exceeding a list's [LINELIMIT] value
error_text_user_block_over	Specify error text returned in some cases of exceeding a user's mailMsgMaxBlocks value or the [BLOCKLIMIT] value for an alias
error_text_user_line_over	Specify error text returned in some cases of exceeding a user's [LINELIMIT] value
header_limit	Sets a maximum size for the primary (outermost) message header
line_limit	Limit the size of messages allowed through the MTA
max_header_block_use	Fine tune message fragmentation
max_header_blocks	Truncate message header after the specified number of MTA blocks
max_header_line_use	Fine tune message fragmentation
max_header_lines	Truncate message header after the specified number of lines
max_mime_levels	Degree to look inside MIME messages during processing
max_mime_parts	Number of parts to look at when processing MIME messages
normal_block_limit	Maximum size of message to treat as being of normal or higher priority
non_urgent_block_limit	Maximum size of message to treat as being of non-urgent priority
second_class_block_limit	Maximum size of message to treat as being of second class priority
urgent_block_limit	Maximum size of message to treat as being of urgent priority
Message Store insertion logging MTA options	
logfile	logfile options set at the MTA level (mta.logfile.*option-name* in Unified Configuration) affect the MTA's logging of insertion of messages into the Message Store; that is, affect logging of message insertions performed by ims-ms and tcp_lmtpss_* channels

Message tracking MTA options	
log_times	(New in MS 8.0) Include requested deferral time in MTA message transaction log entries
log_tracking	(New in MS 8.0) Include tracking ID in MTA message transaction log entries
tracking_hash_algorithm	(New in 8.1.0.3) Select the hash algorithm the MTA uses to generate hashes of tracking and recall secrets. The default is SHA-1.
tracking_mode	(New in MS 8.0) Enable/disable the MTA's tracking support
tracking_debug	(New in MS 8.0) Level of debug output regarding tracking
tracking_retries	(New in MS 8.0) How many times the MTA will reattempt a tracking update
tracking_retry_delay	(New in MS 8.0) The time to wait between tracking update reattempts
MeterMaid MTA options	
enable_sieve_metermaid	(New in MS 8.0) Control whether Sieve filters may use the private metermaid extension
metermaid_backoff	(New in MS 7.2)
metermaid_expire	(New in MS 7.2)
metermaid_host	(New in MS 7.2)
metermaid_port	(New in MS 7.2)
metermaid_secret	(New in MS 7.2)
metermaid_timeout	(New in MS 7.2)
MLS MTA options	
error_text_mls_access_failure	(New in MS 7.0) RESTRICTED: Not yet used
ldap_mlsrange	(New in MS 7.0) RESTRICTED: Not yet used
mls	(New in MS 7.0) RESTRICTED: Not yet fully implemented
MTQP MTA options	
mtqp_expire	(New in MS 8.0) Specify the MTQP expiration, in seconds
mtqp_port	(New in MS 8.0) Specify the MTQP port
mtqp_timeout	(New in MS 8.0) Specify the MTQP timeout, in seconds
Notification message MTA options	
bounce_block_limit	Limit the amount of original message content included in bounce messages
content_return_block_limit	Force NOTARY non-return of content flag for messages over the specified size

history_to_return	Control the amount of delivery attempt history included in bounced messages
ldap_domain_attr_report_address	Name of domain level LDAP attribute whose value specifies a domain-specific postmaster address
lines_to_return	Lines included when returning samples of message content (as in warning messages)
notary_decode	Control whether encoded-words in the original message header are decoded when performing a %H substitution during the MTA's generation of DSNs and MDNs
notary_quote	Specify the character that marks substitution sequences in return_*.txt files and disposition_*.txt files
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
return_verify	(New in MS 7.0.5, replacing former imta_return_verify MTA Tailor option) Enable shell script logging within the MTA return job
use_precedence	Control whether delayed delivery notification messages are sent for list and bulk precedence messages
use_warnings_to	DELETED as of MS 7.0.5: Whether to send DSNs to Warnings-to: address
Password and TLS MTA options	
plaintextmincipher	(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
smtpproxypassword	(New in MS 7.0.5) Replaces the former TCP/IP channel option file option PROXY_PASSWORD
sslnicknames	(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	
defer_group_processing	Set a default for whether direct LDAP group (list) expansion is deferred to the reprocess channel

log_mtpriority	(New in MS 8.0) Include message MT-PRIORITY in MTA message transaction log entries
log_priority	Include message priority in MTA message transaction log entries
mtpriority_policy	(New in MS 8.0) MT-PRIORITY policy name to announce in SMTP EHLO response
normal_block_limit	Maximum size of message to treat as being of normal or higher priority
non_urgent_block_limit	Maximum size of message to treat as being of non-urgent priority
second_class_block_limit	Maximum size of message to treat as being of second class priority
urgent_block_limit	Maximum size of message to treat as being of urgent priority
use_precedence	Control whether delayed delivery notification messages are sent for list and bulk precedence messages
Received header line MTA options	
held_sndopr	Send operator or syslog messages when messages are HELD
id_domain	Set the domain name used in constructing message IDs
max_local_received_lines	Occurrences of the local host name in Received: headers after which a message will be HELD
max_mr_received_lines	Number of MR-Received: headers after which a message will be HELD
max_received_lines	Number of Received: headers after which a message will be HELD
max_total_received_lines	Number of Received:, MR-Received: or X400-Received: headers after which a message will be HELD
max_x400_received_lines	Number of X400-Received: headers after which a message will be HELD
received_domain	Specify the domain name (identifying the system itself) to use in constructing Received: headers
received_version	Specify the IMTA version string to use in constructing Received: header lines; use is **NOT RECOMMENDED**
Sieve filter MTA options	
systemfilter	System Sieve filter, applied to all messages at each enqueue
sieve_body_needed	Specifies whether or not to retain decoded MIME body information when initial message analysis is performed.

sieve_mime_needed	Specifies whether or not to retain inner MIME structure and header information when initial message analysis is performed.
Sieve filter interpretation MTA options	
decode_encoded_words	RESTRICTED: Decode encoded words during Sieve processing
defer_header_addition	(New in MS 7.0.5.30.0) Control whether Sieve filters see any headers added prior to Sieve processing
filter_discard	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 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 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.e.*, maximum addresses that will be used from a redirect :list Sieve action
max_redirects	Maximum number of Sieve "redirect" actions (*i.e.*, 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_list_size	(New in MS 8.1) Maximum number of iterations allowed in a Sieve :matches operation

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 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 MS 6.2) Specify the size of the per-process cache of tokenized Sieve filters
filter_cache_timeout	(New in MS 6.2) Specify the retention time for entries in the per-process cache of tokenized Sieve filters
Sieve language extension MTA options	
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_redis	(New in MS 8.0.2.3) Control whether Sieve filters may use the private redis extension
enable_sieve_regex	Control whether Sieve filters may use the regex extension (the :regex 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 encounted 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_dkim	(New in 8.1.0.6) Include information about DKIM signing operations in message transaction log entries
log_filter	Include applicable Sieve filter actions in MTA message transaction log entries
log_from	(New in 8.1.0.2) Include the address found in the From: header field in message transaction log entries
log_smartsend	(New in 8.1.0.1) Include additional information about smartsend plugin 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_spamfilter*N*_error	Specify default error text to return in cases of spam/virus filter package problems

ldap_domain_attr_optin*N*	Name of domain-level LDAP attribute selecting opt-in to spam/virus filter package *N* processing
ldap_optin*N*	Name of user-level LDAP attribute selecting opt-in to spam/virus filter package *N* processing
ldap_source_optin*N*	Name of user-level LDAP attribute selecting source opt-in to spam/virus filter package *N* processing
ldap_optoutN	Name of user-level LDAP attribute used to opt-out of 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 MS 6.3, obsoleted in favor of the new-in-MS-6.3 spamfilter1_* options with which they are (became in MS 6.3) synonymous
spamfilter*N*_action_*M*	(Values for *N* of 1--4 are new in MS 6.2; values of 5--8 are new in 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 spamfilter*N*_verdict_*M* verdict
spamfilter*N*_config_file	(Values for N of 1--4 are new in MS 6.2; values of 5--8 are new in MS 6.3.) Location of the configuration file for the Nth spam/virus filter package
spamfilter*N*_final	(Values for N of 1--4 are new in MS 6.2; values of 5--8 are new in MS 6.3.) Control whether the MTA passes the "intermediate" *vs*. "final" address form to the Nth spam/virus filter package
spamfilter*N*_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
spamfilter*N*_library	(Values for N of 1--4 are new in MS 6.2; values of 5--8 are new in MS 6.3.) Location of the client shared library for the Nth spam/virus filter package
spamfilter*N*_name	(New in MS 7.0.5) Specify a symbolic name for the *N*th spam/virus filter package.
spamfilter*N*_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
spamfilter*N*_null_optin	(Values for *N* of 1-4 are new in MS 6.2; values of 5--8 are new in MS 6.3.) Specify a value for the attribute named by the ldap_domain_attr_optin*N* MTA options, the ldap_optin*N* MTA options, the (new in MS 8.0.1.3) ldap_optout*N* MTA options, or the (new in MS 6.3) ldap_source_optin*N* MTA options that means that the attribute should be ignored; the purpose is to allow use of these attributes by provisioning facilities that find it easier

	to set an attribute value rather than to remove an attribute entirely
spamfilter*N*_optional	What to do if a spam/virus filter package is not responding: temporarily reject message *vs.* allow to pass unfiltered, plus whether to send syslog notice
spamfilter*N*_received	(New in MS 6.2) Specify whether or not to generate a pseudo-Received: header line to pass to the spam/virus filter package.
spamfilter*N*_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
spamfilter*N*_string_action	(Values for N of 1--4 are new in MS 6.2; values of 5--8 are new in 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
spamfilter*N*_verdict_*M*	(Values for N of 1--4 are new in MS 6.2; values of 5--8 are new in 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_hash_algoritm	(New in 8.1.0.3) Hash algorithm to use when building 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
log_sndopr	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
log_syslog_prefix	(New in MS 8.0.2.3.) Specifices the prefix used on MTA message transaction log file entries sent to syslog
sndopr_prefix	(New in MS 8.0.2.3.) Set the prefix attached to syslog notices.
sndopr_priority	Set the priority of operator broadcast or the syslog level of syslog messages
spamfilter*N*_optional	What to do if a spam/virus filter package is not responding: temporarily reject message *vs.* 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)
log_auth	(New in MS 7.0.5) Include SMTP MAIL FROM's AUTH parameter in MTA message transaction log entries
log_callout_delays	(New in MS 8.0) Include timers showing the time for responses from external components in MTA message transaction log entries
log_connection	Include connection information in MTA transaction log entries
log_connections_syslog	Send MTA connection transaction log file entries to syslog (UNIX) in addition to, or instead of to, MTA connection transaction log file
log_conversion_tag	(New in MS 7.0.5) Include conversion tag(s) field in MTA message transaction log entries
log_debug	RESTRICTED: Enable debugging of MTA logging operations

log_deq	(OpenVMS only) Specify the default extend quantity for the MTA transaction log file(s)
log_delivery_flags	(New in MS 7.0.5) Include delivery flags field in MTA message transaction log entries
log_diagnostics	(New in MS 7.0u1) Include diagnostics in MTA message transaction log entries
log_dkim	(New in 8.1.0.6) Include information about DKIM signing operations in message transaction log entries
log_envelope_id	(New in MS 6.1) Include envelope id in MTA message transaction log entries
log_filename	Include message file names in MTA message transaction log entries
log_filter	Include applicable Sieve filter actions in MTA message transaction log entries
log_format	Control the format of the MTA transaction log file(s)
log_from	(New in 8.1.0.2) Include the address found in the From: header field in message transaction log entries
log_futurerelease	(New in) Include value of FUTURERELEASE SMTP extension in MTA message transaction log entries
log_header	Include message headers in MTA message transaction log entries
log_imap_flags	(New in MS 7.0.5) Include any IMAP flags set by the MTA in MTA message transaction log entries
log_intermediate	(New in MS 6.2) Include the "intermediate" address and/or "original" (RCPT TO: command line) forms of destination address in MTA message transaction log entries
log_isc_status	(New in MS 8.0.2) Include Indexed Search Converter status information in LMTP server MTA message transaction log entries
log_local	Include the local domain name on "bare user name" addresses in MTA message transaction log entries
log_mailbox_uid	(New in MS 7.0.5) Include the IMAP UID and UIDVALIDITY of messages delivered by ims-ms channel to the Message Store in MTA message transaction log entries
log_message_id	Include message IDs in MTA message transaction log entries
log_messages_syslog	Send MTA message transaction log file entries to syslog in addition to, or instead of to, MTA message transaction log file
log_mtpriority	(New in MS 8.0) Include message MT-PRIORITY in MTA message transaction log entries

log_node	(OpenVMS only prior to MS 6.3) Include the node name on which process runs in MTA message transaction log entries
log_notary	Include a NOTARY (delivery receipt) flags indicator in MTA message transaction log entries
log_priority	Include message priority in MTA message transaction log entries
log_process	Include process ID in MTA message transaction log entries
log_queue_time	(New in 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
log_reason	(New in MS 6.3) Include reason for message rejection in MTA message transaction log entries
log_remota_mta	(New in MS 8.0.2.3) Include remote MTA name in MTA message transaction log entries
log_sensitivity	Include message's sensitivity value in MTA message transaction log entries
log_smartsend	(New in 8.1.0.1) Include additional information about smartsend plugin actions in MTA message transaction log entries
log_sndopr	Send an operator or syslog message if the MTA's logging facilities encounter a difficulty
log_syslog_prefix	(New in MS 8.0.2.3.) Specifices the prefix used on MTA message transaction log file entries sent to syslog
log_times	(New in MS 8.0) Include requested deferral time in MTA message transaction log entries
log_tracking	(New in 8.0) Include message tracking ID in MTA message transaction log entries
log_transactionlog	(New in MS 8.0) Include Sieve "transactionlog" action strings in MTA message transaction log entries
log_uid	(New in MS 8.0) Include recipient UIDs in MTA message transaction log entries
log_use_xtext	(New in MS 8.0) Controls xtext encoding of addresses in MTA message transaction log entries
log_username	Include the username for an enqueuing process in MTA transaction log entries
log_8bit_encode	Controls how non-ASCII characters are written in XMl format MTA transaction log file(s)
return_cleanup_period	Run site's cleanup script every *N*th run of return_job
return_split_period	Start new mail.log_current file every *N*th run of return_job

separate_connection_log	Write connection transaction log entries to a separate file than message transaction log entries
OpenVMS user agent MTA options	
delivery_receipt_off	(OpenVMS only) Comment string that disables delivery receipt request
delivery_receipt_on	(OpenVMS only) Comment string that enables delivery receipt request
dis_nesting	(OpenVMS only) Nesting allowed for VMS MAIL @DIS lists
form_names	(OpenVMS only) List of the names of pop-up form images
mail_delivery_filename	(OpenVMS only) Set MAIL.DELIVERY filename
missing_address	(OpenVMS only) Address to insert of VMS MAIL From: if empty
multinet_mm_exclusive	(OpenVMS only) VMS MAIL *vs.* Multinet MM mailbox
read_receipt_off	(OpenVMS only) Comment string that disables read receipt request
read_receipt_on	(OpenVMS only) Comment string that enables read receipt request
safe_tcl_mode	(OpenVMS only) Control handling of Safe-Tcl message parts
use_mail_delivery	(OpenVMS only) Enable MAIL.DELIVERY processing
vms_mail_exclusive	(OpenVMS only) VMS MAIL *vs.* 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.

52.6 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.

52.7 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.)

52.7.1 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 52.3 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.

52.7.2 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 -- effectively a "domain catchall address" 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 52.4 alias_domains MTA option bit values

Bit	Value	Usage
0	1	Look up *localpart*. Clearing this bit disables the lookup of unadorned localparts for channels other than the local channel; for the local channel, localparts are always looked up.
1	2	Look up *localpart@domainname*.
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 * character for the username; more precisely, an *+*@*domainname* (if the address included a subaddress), and an *@*domainname* 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 for aliases, at which point alias_domains can affect aliasing of addresses rewriting to those other channels. Note that the effect of bit 2 (value 4), that is, the probes with the * character as the localpart, can be controlled on a per-channel basis via the aliaswild channel option.

52.7.3 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
```

The alias_magic MTA option can be overriden for specific channels via the aliasmagic channel option.

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.

52.7.4 Alias and address MTA options: alternate_recipient (string)

(New in MS 8.0.1.) The alternate_recipient MTA option specifies the comment string, including the surrounding parentheses, that is used to specify an alternate recipient address as part of a mailing list address entry. The default value for this string is (ALTERNATE-RECIPIENT).

For example, assuming the default value of this option, an entry of the form:

```
listmember@domain.com (alternate-recipient listalternate@domain.com)
```

would associated the alternate address listalternate@domain.com with the mailing list address listmember@domain.com.

52.7.5 alternate_recipient_mode Option

The alternate_recipient_mode MTA option controls the order in which additional alternate recipients are added to an existing alternate recipient list. Possible values are:

0 old recipients follow new recipient

1 new recipient follows old recipient

2 new recipient replaces any old recipients

3 new recipient is silently dropped if any old recipients are present

The default is 0, which is consistent with military messaging requirements.

52.7.6 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, |.

52.7.7 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.

52.7.8 idn_config_file Option

(New in MS 8.0.2.) The idn_config_file MTA option specifies the location of an optional IDNKIT configuration file. Note that this file should not be necessary in normal usage. The option has no default value.

52.7.9 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.

52.7.10 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

addressses 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 52.5 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
```

52.7.11 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.

52.7.12 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".

52.7.13 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 MS 6.2, it was equivalent to 2, meaning that envelope To addresses are placed in a To: header line. As of MS 6.2, it is equivalent to 1, meaning to pass such messages through unchanged, in accordance with what RFC 5322 now recommends.

Table 52.6 `missing_recipient_policy` option values

Value	Action
0	Default; in 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.e.*, ;) To: header line. The phrase used in the group construct is controlled by the `missing_recipient_group_text` 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 `acceptalladdresses` channel option, if used, modifies the timing and form of the rejection.)

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`.

52.7.14 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.

52.7.15 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.

52.7.16 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.

52.7.17 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, =.

52.7.18 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.

52.7.19 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.

52.7.20 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 52.7 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, |) *source-channel*|*from-address*|*to-address* 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, |) *source-channel*|*from-address*|*to-address* 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.

52.7.21 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.

52.7.22 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 cannot 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 52.8 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, |) *source-channel|destination-channel|address*
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, |) *source-channel|destination-channel|address*
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, |) *destination-channel|address*
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, |) *destination-channel|address*
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).)

Note that as of 8.0.1.3, the `usereversedatabase` source channel option can be used to override the setting of `use_reverse_database` on a channel by channel basis.

52.7.23 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.

52.8 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 autoresponse 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 autoresponse 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`.

52.8.1 Autoresponse 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 sytem-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 sytem-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).

52.8.2 Autoresponse periodicity MTA options: `notify_maximum_timeout` (integer)

(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.

52.8.3 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.

52.8.4 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.

52.8.5 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.

52.8.6 Autoresponse periodicity MTA options: vacation_hash_algorithm (hash algorithm name)

The vacation_hash_algorithm MTA option controls what hash algorithm the MTA uses to generate the names for vacation database entries. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, SHA256, SHA512, MD128, or MD160. The default if this option is not specified is to use MD4 for hashing the autoreply attributes and MD5 to rehash the name if it is too long. Note that the setting of this option must be the same across a deployment for successful coordination of vacation responses across hosts.

52.8.7 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.

52.8.8 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.

52.8.9 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:///DATAROOT/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 overidden 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.

As of MS 8.1.0.1, Redis URLs are also supported. These have the same semantics as memcache URLs, so a sensible setting would be:

```
redis:///$U@$2I
```

52.9 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.

52.9.1 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.

52.9.2 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`).

52.10 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`

52.11 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.

52.11.1 `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
```

52.12 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.

52.12.1 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.

52.12.2 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.

52.12.3 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.

52.12.4 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_frustation_limit value, which by default is 100, then that process will no longer even attempt counter update operations.

52.12.5 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.

52.12.6 Syslog MTA options: log_sndopr (bitmask)

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. New in MS 8.0, the option takes a bit-encoded value, with bit 0 (value 1) and bit 1 (value 2) having meaning: bit 0 enables syslog notices regarding trouble writing transaction log entries; bit 1 enables syslog notices regarding trouble creating or updating channel counters.

The sndopr_priority option controls the syslog level (facility and severity) of the syslog messages generated.

52.12.7 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.

52.13 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, and reverse_database_url, which can specify either Memcache (new in MS 8.0.2.3) Redis URLs for storing MTA database data;
- general controlling the case sensitivity of general database lookups.

See also the Direct LDAP MTA options, the Autoresponse periodicity MTA options, and the MeterMaid MTA options.

52.14 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 option or debug Dispatcher option.

For a quick view of whether/what debugging you may have enabled, try:

```
msconfig> show *debug*
```

52.14.1 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 enqueuing 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.

52.14.2 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.)

52.14.3 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.

52.14.4 Debug MTA options: debug_flush (0 or 1)

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 Dispatcher debug output.

52.14.5 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.

52.14.6 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.

52.14.7 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.

52.14.8 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 MS 6.2) debugging regarding the reason when a message for multiple recipients is "split up" into different copies, and (as of MS 6.3) debugging regarding SRS/MUL decoding; 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.

52.14.9 Debug MTA options: os_debug (0 or 1)

RESTRICTED. The 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 dequeueing messages. When enqueuing 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.

52.14.10 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.)

52.14.11 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 MS 6.1p1 and later will appear in the IMTA_LOG:return-*uniquestring*.log log file.

52.14.12 Debug MTA options: return_verify (0 or 1)

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.

52.14.13 symbiont_debug Option

DEPRECATED.

Formerly, the symbiont_debug MTA option could be used to enable debugging of the PMDF Process Symbiont.

52.14.14 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.

52.15 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 connnecting 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.

52.15.1 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).

52.15.1.1 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`.

52.15.1.2 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.

52.15.1.3 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`.

52.15.1.4 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`.

52.15.1.5 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).

52.15.1.6 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 52.9 ldap_use_async MTA option bits

Bit	Value	Usage
0	1	ldap_group_url1 (mgrpDeliverTo) URLs
1	2	ldap_group_url2 (memberURL) URLs
2	4	ldap_group_dn (uniqueMember) DNs and as of Messaging Server 7.0.5, also ldap_group_dn2 DNs
3	8	[AUTH_LIST], [MODERATOR_LIST], [SASL_AUTH_LIST], [SASL_MODERATOR_LIST] list named parameter URLs (legacy configuration), or in Unified Configuration, alias_auth_list, alias_moderator_list, alias_sasl_auth_list, alias_sasl_moderator_list, alias option URLs
4	16	[CANT_LIST], [SASL_CANT_LIST] list named parameter URLs (legacy configuration), or in Unified Configuration, alias_cant_list, alias_sasl_cant_list 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.

52.15.1.7 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).

52.15.1.8 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.

52.15.2 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_schema1`, `ldap_domain_filter_schema2` and `ldap_domain_root`.
- Options to rename which LDAP attributes are used/recognized in domain entries are discussed under Direct LDAP attribute name MTA options; see especially the `ldap_domain_attr_*` MTA options.

52.15.2.1 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 submisssions from the Internet) or from channels "internal" to the MTA (such as reprocess or conversion channels), reject the submission attempt with a temporary error (4*yz* error) so that the message remains where it was (on the external, Internet host, or on the internal channel) which can re-attempt submisssion 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 validiation 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.

52.15.2.2 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!**

52.15.2.3 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 52.10 `domain_uplevel` option bit values

Bit	Value	Usage
0	1	When set, domain map searches, such as the `$V` search in the typical configuration's `$*` 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 `service.defaultdomain` (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 of 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.

See also the `ldap_domain_attr_uplevel` MTA option for specifying the name of a domain-level LDAP attribute which allows some domain-level override of the MTA-wide setting of `domain_uplevel`.

52.15.2.4 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.

52.15.2.5 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.

52.15.2.6 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.

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

(New in 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.

52.15.2.8 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.

52.15.2.9 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)
```

52.15.2.10 `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.* building and requesting a list of "known" domain attributes. The default of `0` 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.

The "known" attribute list consists of all attributes specified in the various `ldap_domain_attr_*` and similar MTA options, as well as any attributes specified in any LDAP domain map attribute substitutions. This should cover all the cases where the MTA requests domain attributes via its internal domain lookup facilities. However, in the unlikely event that other calls are made to the domain map facility, a site that has added additional domain attributes may be forced to use the default setting of `0` so that LDAP domain queries will return those additional, custom LDAP attribute values.

52.15.2.11 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`.

52.15.2.12 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.

52.15.2.13 `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 host name 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.

An important use of the ldap_host_alias_list is in email configurations spanning multiple data centers, with routing via SMTP between data centers and delivery LMTP within each data center. In such a setup the mailbox delivery_options MTA option clause is set for LMTP but is left as mailhost-dependent, rather than as in a typical configuration being marked with "#" to be mailhost-independent. Then all the LMTP hosts for the local data center are listed in the ldap_host_alias_list MTA option value on each MTA. Note that as the number of stores becomes large it's preferable to use a naming convention for stores along with a wildcard ldap_host_alias_list value, to avoid configuration churn as stores are added or deleted.

52.15.2.14 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 "l" 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 "l" 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).

52.15.3 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.

52.15.3.1 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 ("l") 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.ugldapport`, or in Unified Configuration the `ugldaphost` and `ugldapport` 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.

52.15.3.2 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.

52.15.3.3 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 address 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 or (in 7.0.4 or laster) space-separated attribute names may be specified.

In unifified configurations the local.imta.mailaliases configutil parameter is equivalent to ldap_mail_aliases; in legacy configurations the two are separate parameters and the ldap_mail_aliases MTA option, if specified, overrides the local.imta.mailaliases configutil option.

Of neither option 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".

52.15.3.4 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.

52.15.3.5 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`.

52.15.3.6 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 configutation, 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.

52.15.4 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.

52.15.4.1 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`.

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

(New in 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.

52.15.4.3 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)
```

52.15.4.4 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.

52.15.4.5 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.

52.15.4.6 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.

52.15.4.7 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.

52.15.4.8 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; the two options are equivalent in unified 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 sasl.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).

52.15.4.9 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 `sasl.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).

52.15.4.10 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`.

52.15.5 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.

52.15.5.1 User/group LDAP attribute validity and interpretation options: `aliasdetourhost_null_optin` (string)

(New in 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.

52.15.5.2 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.

52.15.5.3 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 strucute 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.

52.15.5.4 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 table Modifier letters for delivery_options, 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 52.11 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, by forcing to the reprocess channel
^	Inclusive time limit processing - Check for LDAP attributes specifying vacation start and end time; only apply this entry if the current time is after any specified start time and before any specified end time.
%	(New in 7.0.5) Exclusive time limit processing - Check for LDAP attributes specifying vacation start and end time; only apply this entry if the current time is before any specified start time or after any specified end time.
#	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 .HELD 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 or tcp_lmtpcs channel (Message Store delivery channels); a mailDeliveryOption value of native is implemented by routing the address local-part to the legacy 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 MS 6.1 or later system):

```
#*mailbox=@$X.LMTP:$M%$\$2I$_+$2S@lmtpcs-daemon,
&members=*,
*native=$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
```

where that assumes the use of rewrite rules along the lines of

```
.LMTP   $E$F$U%$H.LMTP@lmtpcs-daemon
.LMTP   $B$F$U%$H@$H@lmtpcs-daemon
```

and an outbound `tcp_*` channel, (typically but not necessarily named `tcp_lmtpcs`), corresponding to the `lmtpcs-daemon`; official channel host name, and where the channel is marked with the `multigate` channel option.

For another example of an alternate setting of `delivery_options`, see Additional `ims-ms` channels.

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.

52.15.5.5 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 MS 6.3, but prior to 7.4-18.01) was:

```
ldap:///$A??sub?mail=*
```

And prior to 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 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 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).)

52.15.5.6 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 52.12 `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.

52.15.5.7 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.

52.15.5.8 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.

52.15.5.9 `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 host name 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.

An important use of the `ldap_host_alias_list` is in email configurations spanning multiple data centers, with routing via SMTP between data centers and delivery LMTP within each data center. In such a setup the mailbox `delivery_options` MTA option clause is set for LMTP but is left as mailhost-dependent, rather than as in a typical configuration being marked with "#" to be mailhost-independent. Then all the LMTP hosts for the local data center are listed in the `ldap_host_alias_list` MTA option value on each MTA. Note that as the number of stores becomes large it's preferable to use a naming convention for stores along with a wildcard `ldap_host_alias_list` value, to avoid configuration churn as stores are added or deleted.

52.15.5.10 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 "l" 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 "l" 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).

52.15.5.11 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 or permanent identifier. (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.

52.15.5.12 Spamfilter MTA options: optin_user_carryover (bitmask)

New in 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.

52.15.5.13 Direct LDAP attribute interpretation MTA options: `process_substitutions` (bit-encoded integer)

New in 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 52.13 `process_substitutions` MTA option bits

Bit	Value	Usage
0	1	If set, enables substitutions in `mgrpDisallowedBroadcaster` (`ldap_cant_url`)
1	2	If set, enables substitutions in `mgrpAllowedBroadcaster` (`ldap_auth_url`)
2	4	If set, enables substitutions in `mgrpModerator` (`ldap_moderator_url`)
3	8	If set, enables substitutions in `mgrpDeliverTo` (`ldap_group_url1`)
4	16	If set, enables substitutions in `memberURL` (`ldap_group_url2`)
5	32	(New in Messaging Server 7.0) If set, enables subaddress `$S` substitution in `mgrpErrorsTo` (`ldap_errors_to`)
6	64	(New in Messaging Server 7.0u3) If set, enables substitutions in `mgrpJettisonBroadcasters` (`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 (`$A`), domain (`$D`), host (`$H`), and local-part (`$L`) are all derived from the authenticated sender address. In the case of list expansion attributes, all of these substitution values are derived from the envelope recipient address that specified the list. In both cases, however, the subaddress substitution (`$S`) is derived from the current envelope recipient address.

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=$S)
```

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.

52.15.5.14 Direct LDAP attribute interpretation MTA options: `route_to_routing_host` (-1, 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.

New in 8.0.1.1, a setting of -1, in addition to disabling routing of addresses with nonlocal mailhosts to the mail routing host, acceptance of addresses without a user entry and routing them to the `mailRoutingSmartHost` defined for the corresponding domain is also disabled. The smart host for the domain will continue to be used for addresses that match user entries that fail to specify a mailhost.

52.15.5.15 Sieve filter MTA options: `sieve_user_carryover` (0 or 1)

New in 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.

52.15.5.16 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 intrpreted as follows:

Table 52.14 `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 `spare_N_separator` 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.

52.15.5.17 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.

52.15.5.18 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.

52.15.5.19 Direct LDAP MTA options: `prefix_text_attr` (HTML attribute list)

New in 8.0.1.3. A better-looking result may be produced by associating prefix text inserted into text/html message parts with a particular set of HTML attributes. The `prefix_text_attr` MTA option may be used to do this. If specified, the option causes any prefix material that is inserted into text/html parts to be enclosed in a `<div attrs">` element, where "attrs" is the value of the option.

The default for this option is the empty string, which suppresses the insertion of any additional elements.

52.15.5.20 Direct LDAP MTA options: `suffix_text_attr` (HTML attribute list)

New in 8.0.1.3. A better-looking result may be produced by associating suffix text inserted into text/html message parts with a particular set of HTML attributes. The `suffix_text_attr` MTA option may be used to do this. If specified, the option causes any suffix material that is inserted into text/html parts to be enclosed in a `<div attrs">` element, where "attrs" is the value of the option.

The default for this option is the empty string, which suppresses the insertion of any additional elements.

52.15.6 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 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 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 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 52.15 MTA LDAP attribute name options

Option	Default attribute name(s)	Valid	Meaning and notes
Per-user/group attributes			
ldap_objectclass	objectClass	UGD	
ldap_user_status	inetUserStatus	U	Prior to Messaging Server 7.0, the supported values 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": ldap_user_status=mailUserStatus ldap_user_mail_status=inetUserStatus
ldap_user_mail_status	mailUserStatus	U	Valid values are active, inactive, disabled, deleted, overquota, hold, removed (new in MS 6.0), defer (new in MS 6.3), defer-submit (new in MS 6.3), deliver (new in 7.3-11.01), and deliver-disabled (new in 8.0.1.3/8.0.2.1). A status of removed is equivalent to deleted from the MTA's point of view; it exists as a distinct status for the benefit of the commcli user purging operation. The statuses defer and defer-submit tell the MTA 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 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. A status of deliver is treated by the MTA as active 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. Finally, a status of deliver-disabled is treated by the MTA as disabled by as active by other components.
ldap_group_status		G	Prior to Messaging Server 7.0, the supported values were a strict subset of the values supported for inetMailGroupStatus; in particular, the supported values were active, inactive, and deleted. New in Messaging Server 7.0, 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.
ldap_group_mail_status	inetMailGroupStatus	G	Supported values are active, deleted, removed, disabled, hold, inactive, ew in Messaging Server 7.0 defer anddefer-submit, and new in MS 8.0.1.3/8.0.2.1 deliver-disabled.
ldap_permid	uid	UG	As of Messaging Server 8.0.2, the attribute specified by the ldap_permid option is used for construction of user and group identifiers rather than the attribute specified by the ldap_uid option, assuming they differ. The MTA checks that there is only one such attribute and value and that the value is no more than 128 octets long.
ldap_uid	uid	UG	As of MS 6.2, the MTA checks that there is only one such attribute; as of 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.)
ldap_mlsrange		UG	(New in Messaging Server 7.0?) RESTRICTED
ldap_capture		UG	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 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 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 ;format-report selects the usual DSN encapsulated format, whereas the tag ;format-journal selects the envelope "journal" format.
ldap_recipientlimit		UG	Specify an attribute used to store a sending-user-specific maximum number of envelope recipients (additional recipients are rejected), analogous to the recipientlimit channel option. New behavior in 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 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_recipientcutoff		UG	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 recipientcutoff channel keyword. New behavior in 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 MS 63, 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.
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 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 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_optin*N*		UG	(New in 6.3) Sending user analogue of ldap_optin*N* option
ldap_preferred_language	preferredLanguage+	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 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 MS 6.3, the only attributes receiving this treatment are ldap_autoreply_subject (normally mailAutoReplySubject), ldap_autoreply_text (normally mailAutoReplyText), ldap_autoreply_text_internal (normally mailAutoReplyTextInternal), 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 preferredLanguage treatment; and as of Messaging Server 7.3-11.01, the attribute named by ldap_add_tag also receives preferredLanguage treatment.
ldap_preferred_country		UG	(New in MS 6.3-0.15)
ldap_nosolicit		UG	New in 6.2. Specifies solicitation strings used by the SMTP NO-SOLICITING extension that this user doesn't want to receive.
ldap_routing_address	mailRoutingAddress	UG	Used to specify an address to which to route, overriding (as of MS 6.0) the usual mailHost check and mailDeliveryOption interpretation.
ldap_delivery_option	mailDeliveryOption+	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 MS 6.2. Specify an attribute whose value will be applied as a conversion tag for messages coming from this user.
ldap_sender_sieve		UG	(New in MS 8.0.1)
ldap_primary_address	mail	UG	
ldap_alias_addresses	*varies with the schema tag;* mailAlternateAddress *for* ims50 *or* nms41; rfc822mailalias *for* sims40	UG	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.
ldap_equivalence_addresses	mailEquivalentAddress	UG	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). 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 spamfilterN_null_optin MTA options) 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 *Sun Schema Reference Manual* recommends using the `mailAntiUBEService` attribute.
`ldap_optin`*`N`*		UG	(New in MS 6.2.) The presence in a user entry of the attribute named by this option normally (but see the `spamfilter`*`N`*`_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 range from 1 to 8. Note that the *Sun Schema Reference Manual* recommends using the `mailAntiUBEService` attribute for optin use.
`ldap_optout`*`N`*		UG	(New in MS 8.0.1.3.) The presence in a user entry of the attribute named by this option normally (but see the `spamfilter`*`N`*`_null_optin` MTA option) causes messages addressed to this user to be "opted-out" of virus/spam filter package processing (by virus/spam filter package `#` *`N`*). The value of *`N`* can range from 1 to 8.
`ldap_presence`		UG	RESTRICTED: Not yet used.
`ldap_autosecretary`		UG	RESTRICTED: Not yet used.
`ldap_alternate_recipient`		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.
`ldap_start_date`	`vacationStartDate+`	UG	The value for this attribute should have the format *`YYYYMMDDHHMMSSZ`*, which note is in the GMT timezone. The value for this 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 this attribute and inclusive limit processing is in effect, or before the specified limit if exclusive time limit processing is in effect. No start date limit is enforced if this attribute is missing.
`ldap_end_date`	`vacationEndDate+`	UG	The value for this 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 this attribute and inclusive limit processing is in effect, or after the specified limit if exclusive time limit processing is in effect. No end date limit is enforced if this attribute is missing.
`ldap_conversion_tag`	`mailConversionTag`	UG	
`ldap_detourhost_optin`		UG	Opt-in to "detour" routing, as specified by the `aliasoptindetourhost` source channel option
`ldap_blocklimit`	`mailMsgMaxBlocks`	UG	The maximum size, in MTA blocks (see the `block_size` MTA option), of message that may be sent to a user. New in 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.
`ldap_mailhost`	`mailHost`	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 *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: `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).
`ldap_disk_quota`	`mailQuota`	U	Subsequent to the initial release of MS 6.2, support was added for `mailQuota` values specified in units other than bytes; that is, suffix characters K (kilobytes), M (megabytes), and G (gigabytes) are supported
`ldap_message_quota`	`mailMsgQuota`	U	
`ldap_program_info`	`mailProgramDeliveryInfo+`	UG	
`ldap_delivery_file`	`mailDeliveryFileURL`, `mailDeliveryFile`	UG	
`ldap_spare_1`		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a `$E1` substitution.
`ldap_spare_2`		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a `$E2` substitution.

ldap_spare_3		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a $E3 substitution.
ldap_spare_4		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a $E4 substitution as well as in Sieve extlists callouts. Note that new in 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 Accept-Language:, or in the absence of such header lines will use the preference noted in the envelope From user's ldap_preferred_language (normally preferredLanguage) attribute's value.
ldap_spare_5		UGD	Specify an attribute that may be then be accessed in LDAP URL lookups via a $E5 substitution, as well as in Sieve extlists callouts. As of 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 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.
ldap_spare_6		UGD	(New in 7.0-3.01) Specify an attribute that may be then be accessed in LDAP URL lookups via a $E6 substitution, as well as in Sieve extlists callouts. 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.
ldap_autoreply_mode	mailAutoReplyMode+	UG+++	Supported values for this attribute are echo and reply. These modes will appear in a Sieve script as nonstandard :echo and :reply arguments to the vacation action. echo will produce a "processed" message disposition notification (MDN) that contains the original message as returned content. reply will produce a pure reply containing only the reply text. An illegal value won't manifest as any argument to the vacation action and this will produce an MDN containing only the headers of the original message.
ldap_autoreply_subject	mailAutoReplySubject+	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 :subject argument to the vacation action. As of MS 6.2p2, 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. New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form $<attribute><part> where $<attribute> is "B" for the start (beginning) date or "E" for the end date, and <part> is one of the date parts defined in RFC 5260 section 4.2. Note that new in 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 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.
ldap_autoreply_text	mailAutoReplyText+	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 MS 6.2p2, 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. New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form $<attribute><part> where $<attribute> is "B" for the start (beginning) date or "E" for the end date, and <part> is one of the date parts defined in RFC 5260 section 4.2. Note that new in 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 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.
ldap_autoreply_text_internal	mailAutoReplyTextInternal+	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

			`ldap_autoreply_text` MTA option. As of MS 6.2p2, 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. New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form `$<attribute><part>` where `$<attribute>` is "B" for the start (beginning) date or "E" for the end date, and `<part>` is one of the date parts defined in RFC 5260 section 4.2. Note that new in 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 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.
`ldap_autoreply_addresses`		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 `:addresses` argument for the Sieve `vacation` action.
`ldap_autoreply_timeout`	`mailAutoReplyTimeOut+`	UG+++	This attribute stores the duration, in hours, for successive vacation (autoreply) responses to any given mail sender. Used only when `mailAutoReplyMode=reply`. 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 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 `ldap_domain_attr_autoreply_timeout` MTA option) if the domain has its own timeout, or otherwise from the `autoreply_timeout_default` MTA option.
`ldap_filter`	`mailSieveRuleSource+`	UG	This attribute stores a per-user Sieve filter.
`ldap_parental_controls`		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 filtering be applied to this user's (or group's) messages. Any of the values `Yes`, `1`, or `true` 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 `mailSieveRuleSource`. 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 `mail`.)
`ldap_forwarding_address`	`mailForwardingAddress+`	UG	Address(es) used in the expansion of named delivery options with the special value "*" (normally the name "`forward`" is associated with this value).
`ldap_list_id`	`mgrpUniqueId`	G	(New in 7.3-11.01) Single valued attribute specifing 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 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`	`mailDeferProcessing`	UG	This attribute allows per-group or per-list override of the `defer_group_processing` MTA option. Valid values are "`Yes`", "`No`", or (new in MS 6.3p1) "`AFTER_AUTH`". "`AFTER_AUTH`" causes LDAP attribute 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 access checks) to the reprocess channel. New in Messaging Server 7.0u3, a channel name (*e.g.*, "`process_special`" or some similar, special, `reprocess_*` or `process_*` 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`	`mgrpJettisonDomain`	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`	`mgrpJettisonBroadcasters`	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`	`mgrpMsgRejectAction`	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: "TOMODERATOR", which if set instructs the MTA to redirect any access

			failures to the moderator specified by the mgrpModerator attribute. The default (and any other value of this attribute) causes an error to be reported and the message rejected.
ldap_reject_text	mgrpRejectText, mgrpMsgRejectText	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	mgrpBroadcasterPolicy	G	Specifies level of authentication needed to send to the group. Possible tokens are "SMTP_AUTH_REQUIRED" or "AUTH_REQ", both of which mean that the SMTP AUTH command must be used to identify the sender in order to send to the group and any address produced by authentication will be used in subsequent authentication checks, "SMTP_AUTH_USED" or "AUTH_USED", both of which force the use of any authenticated address in authorization checks but does not actually require authentication, "PASSWORD_REQUIRED", "PASSWD_REQUIRED", or "PASSWD_REQ", all of which mean the password to the list specified by the mgrpAuthPassword attribute (see below) must appear in an Approved: header field in the message, "OR", which changes the or_clauses MTA option setting to 1 for this list, "AND", which changes the or_clauses MTA option setting to 0 for this list, and "NO_REQUIREMENTS", which is basically a no-op. "OR" and "AND" 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: "LIST_OPEN", "LIST_MEMBERS", "LIST_MODERATE_NONMEMBERS", "LIST_MODERATE_MEMBERS", and "LIST_MODERATE". Also new in 7.3-11.01, multiple attributes can now be mapping to this attribute slot.
ldap_cant_url	mgrpDisallowedBroadcaster	G	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.
ldap_auth_url	mgrpAllowedBroadcaster	G	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 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 MS 6.3 or later. New in MS 6.3, this now checks for address aliases (*e.g.*, mailAlternateAddress and mailEquivalentAddress) as well as for the canonical address (normally the mail attribute value).
ldap_cant_domain	mgrpDisallowedDomain	G	Domains not allowed to submit messages to this group. A match means access checking has failed and all subsequent checks are bypassed. In MS 6.0 this check is bypassed if the submitter has already matched an ldap_auth_url. Can be multivalued and as of MS 6.2 glob-style wildcards are allowed.
ldap_auth_domain	mgrpAllowedDomain	G	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 MS 6.0 this check is bypassed if the submitter has already matched an ldap_auth_url. As of 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.
ldap_maximum_message_size	mgrpMsgMaxSize	G	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.
ldap_auth_password	mgrpAuthPassword	G	Specifies 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 (see above) and checked against the Approved: field once the header is available. The Approved: field was removed from the header once the checkis complete. But this did not allow for routing to the moderator in the event of a password check failure. 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. The `or_clauses` MTA option acts on this attribute in the same way it acts on the other access check attributes.
`ldap_moderator_url`	`mgrpModerator`	G	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 DNs 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.
`ldap_group_last_access_time`		G	(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`". 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>`".)
`ldap_group_url1`	`mgrpDeliverTo`	G	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 `process_substitutions` MTA option is set in 6.3 or later. See also the `ldap_url_result_mapping` MTA option; with it, a mapping table can be used to manipulate the value(s) of the `ldap_group_url1` attribute.
`ldap_group_url2`	`memberURL`	G	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 `process_substitutions` MTA option is set in 6.3 or later. See also the `ldap_url_result_mapping` MTA option; with it, a mapping table can be used to manipulate the value(s) of the `ldap_group_url2` attribute.
`ldap_group_dn`	`uniqueMember`	G	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 `group_dn_template` MTA option. The default value for this option is "`ldap:///$A?mail?sub?(mail=*)`"; `$A` specifies the point where the `uniqueMember` DN is inserted. As of 7.0.5, if a `GROUP_TEMPLATES` 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 `group_dn_template` MTA option value. Multiple values are supported but only one attribute of this type is allowed on any given group. As of 7.0.5 any mapping specified by the attribute named by the `ldap_url_result_mapping` MTA option will also be applied to the results produced by these attributes.
`ldap_group_dn2`		G	(New in 7.0.5.) Like `ldap_group_dn`, 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 sematics.
`ldap_group_rfc822`	`mgrpRFC822MailMember`, `rfc822MailMember`	G	Mail addresses of members of this list. Multiple values are allowed. `rfc822MailMember` is also supported for backwards compatibility with NMS, but only one of these attributes can be used in any given group.

ldap_url_result_mapping		G	(New in 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 ldap_group_url1, ldap_group_url2, and as of 7.0.5, the ldap_group_dn and ldap_group_dn2 MTA options. That is, ldap_url_result_mapping 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: *LDAP-URL*|*LDAP-result* where LDAP-URL is the (literal string) value of the expansion attribute, 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. This mechanism can be used to define groups based on attributes that don't contain proper email address.
ldap_errors_to	mgrpErrorsTo	G	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. 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 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. New in 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".
ldap_delay_notifications	mgrpDelayNotifications	G	(New in 7.0u3.) Should NOTIFY=DELAY be set on list messages? Supported values are yes and no, with the obvious meanings.
ldap_add_header	mgrpAddHeader	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).
ldap_remove_header	mgrpRemoveHeader	G	Specify header fields to remove from messages posted to the list. Only the field name should be specified.
ldap_add_tag	*No default;* mgrpListTag *recommended*	G	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 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.
ldap_prefix_text	mgrpMsgPrefixText	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.
ldap_suffix_text	mgrpMsgSuffixText	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.
ldap_expandable	mgmanMemberVisibility, expandable	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 anyone, 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).
MAILSERV attributes (new in 7.4-18.01)			
ldap_list_name	mgrpListName	G	Name associated with the list for administrative purposes. The value must be a UTF-8 string.
ldap_list_description	mgrpDescription	G	Textual description of the list. UTF-8 string. Multivalued and supports language-tagged attribute values.
ldap_list_advertised	mgrpAdvertised	G	Whether or not to advertise the list in the MAILSERV GUI. Possible values are yes or no.
ldap_list_public_roster	mgrpPublicRoster	G	Should the membership list be visible to anyone, or merely to members, or to admins_only.
ldap_list_subscribe_policy	mgrpSubscribePolicy	G	Policy for handling list subscription requests. Possible values are: • "immediate" - honor subscription requests immediately • "confirm" - send confirmation e-mail • "to_moderator" - require moderator approval • "both" - require both confirmation and moderator approval.
ldap_list_unsubscribe_policy	mgrpUnsubscribePolicy	G	Controls list unsubscribe policy. Same possible settings as for ldap_list_subscribe_policy.
ldap_list_trust_new_members	mgrpTrustNewMembers	G	
Head of household control support attributes			
ldap_hoh_filter	mailSieveRuleSource		(New in MS 6.2.) Specify an attribute to request when performing a head of household filter lookup.
ldap_hoh_owner	mail		(New in MS 6.2.) Attribute in which to find the owner of the HOH Sieve.
Schema 2 support attributes			
ldap_attr_domain1_schema2	sunPreferredDomain++	D	Attribute used to store the primary domain in Schema 2.
ldap_attr_domain2_schema2	associatedDomain++	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	inetDomainBaseDn	D	Domain entry attribute used to store the baseDN of domain entries.
ldap_domain_attr_alias	aliasedObjectName	D	Schema 1 attribute used in domain alias entries to specfify the DN of the actual domain entry.
ldap_domain_attr_uplevel		D	(New in 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 *found*. Thus setting bit 0 (value 1) has no effect unless bit 0 (value 1) 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 *for subdomains of this particular domain* while domain uplevel matching is still possible for other domains.
ldap_domain_attr_canonical	inetCanonicalDomainName	D	Specifies the canonical domain name for domains whose member entries overlap.
ldap_domain_attr_uid_separator	domainUidSeparator	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	mailRoutingHosts	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 rewriting the address, but if the MTA option route_to_routing_host is set to 1 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.
`ldap_domain_attr_smarthost`	`mailRoutingSmartHost`	D	If a user address is not found in the directory, then route onwards, inserting the `mailRoutingSmartHost` value into the address as a source route.
`ldap_domain_attr_status`	`inetDomainStatus`	D	The attribute containing the domain's overall status.
`ldap_domain_attr_mail_status`	`mailDomainStatus`	D	The attribute containing the domain's mail service status. Valid values for this attribute are: `active`, `inactive`, `deleted`, `hold`, `disabled`, `overquota`, and (new in MS 6.0) `unused` and `removed`; other values are interpreted as `inactive`. Note that the `imquotacheck` utility is what updates `mailDomainStatus` to set it to `overquota`.
`ldap_domain_attr_blocklimit`	`mailDomainMsgMaxBlocks`	D	Set a domain limit for how large of message users in the domain may receive. New in 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`	`mailDomainConversionTag`	D	New in 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 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_optin`*N*		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 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 `preferredMailHost` attribute, used in provisioning, that would be one possibly appropriate attribute to also use for this purpose	D	(New in MS 6.3) Specify a default `mailHost` to take effect for all users in this domain who do not have their own explicit `mailHost` set.
`ldap_domain_attr_disk_quota`		D	
`ldap_domain_attr_message_quota`		D	
`ldap_domain_attr_filter`	`mailDomainSieveRuleSource`	D	Attribute used to specify domain-level Sieve filters.
`ldap_domain_attr_sender_sieve`	`mailDomainSenderSieve`	D	(New in MS 8.0.1)
`ldap_domain_attr_report_address`	`mailDomainReportAddress`	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 `postmaster@domain`. 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`	`mailDomainCatchallAddress`	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; `mailDomainCatchallMapping` recommended	D	(New in 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 `$Y` 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 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 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 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 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 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 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 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/mailing 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/mailing list entries. See the delivery_options MTA option for some discussion regarding enabling use of this attribute on group/mailing list entries.

52.15.6.1 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.

52.15.6.2 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.

52.15.6.3 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
```

52.15.6.4 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`, `removed` (new in MS 6.0), `defer` (new in MS 6.3), `defer-submit` (new in MS 6.3), `deliver` (new in 7.3-11.01), and `deliver-disabled` (new in 8.0.1.3/8.0.2.1).

A status of `removed` is equivalent to `deleted` from the MTA's point of view; it exists as a distinct status for the benefit of the commcli user purging operation.

The statuses `defer` and `defer-submit` tell the MTA 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` 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.

A status of `deliver` is treated by the MTA as active `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`.

(Note that the `acceptalladdresses` channel option, if used, modifies the timing and form of recipient rejections due to errors such as "`overquota`" or "`disabled`" status. New in MS 8.0.1.1.0, the `accepttemporaryfailures` channel option, if used, modifies the timing and form of recipient deferrals due to temporary errors such as "`inactive`" status. Note also that the `use_temporary_error` and `use_permanent_error` MTA options, and `usetemporaryerror` and `usepermanenterror` channel options, can alter the interpretation of whether such status errors are considered temporary *vs*. permanent errors.)

52.15.6.5 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.

52.15.6.6 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, (new in Messaging Server 7.0-0.04) defer and defer-submit, and (new in MS 8.0.1.3/8.0.2.1) deliver-disabled.

(Note that the acceptalladdresses channel option, if used, modifies the timing and form of recipient rejections due to "disabled" status.)

52.15.6.7 Direct LDAP attribute name Base options: ldap_permid (LDAP attribute name)

The ldap_permid base option names a user or group LDAP attribute that contains a permanent identifier for the user. The value of such an attribute will be used preferentially as the identifier for the user or group which is used for such purposes as:

- The unique name (or domain-qualified unique name) for the user in the message store. This includes delivery by ims_master or LMTP, as well as mailbox autocreation by popd, imapd or other store tools.
- The canonical user name for authentication purposes.
- The identifier associated with store access connections when using the imsconnutil tool.
- The identifier used when storing (but not when displaying or accessing) IMAP ACLs or shared folder user names.
- May be part of the DN of the user's entry in the LDAP directory.

The use of the permanent identifier in constructing the user's unique mailbox name in the message store means that attempting to change a user's permanent identifier tends to be quite problematic (a change breaks access to the user's old mailbox). So make every attempt to avoid changing the value of this attribute, value; use some other LDAP attribute for values subject to change (such as the user's legal name, display name or login name) and leave the permanent identifier as an arbitrary, immutable identifier.

The attribute specified by the `ldap_permid` option must be indexed in LDAP as it is used for canoncial user identity searches in LDAP.

In most cases, the value of the attribute named by the `ldap_uid` MTA option will be used as the user or group permanent identifier if the attribute named by `ldap_permid` is not present in the user or group entry. However, for canoncial identity search operations, such as those necessary for Cassandra store account auto-creation, the attribute named by `ldap_permid` must be set for the correct LDAP entry to be found so autocreation can proceed.

52.15.6.8 `ldap_extid` Option

The `ldap_extid` base option names a user LDAP attribute which contains an external display identifier for the user for use with the Cassandra Message Store. The external display identifier must be unique within the user's domain. IMAP shared folders and Access Control Lists (ACLs) are internally stored using the identifier from the ldap_permid attribute, but when shared folder owner names or ACLs are changed or sent over IMAP, the external display identifier is used. If the external display identifier does not contain '@', then the domain name will be added for domains other than the default domain.

The Cassandra Message Store caches a copy of the value of this attribute in the 'userid' column of the user table in the ms_mbox meta data keyspace. To refresh the stored value, TBD.

The default value for the `ldap_extid` base option is the value of the `ldap_uid` MTA option and that option defaults to 'uid'.

52.15.6.9 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 used as the user or group identifier if no permanent identifier (specified by the ldap_permid option) is present in the user or group entry. This option also impacts the default login identity for authentication purposes.

Note that although the MTA option `ldap_uid` exists to rename/redirect the attribute used *for some MTA, Store and authentication purposes*, other components of Messaging Server such as some MTA and Message store utilties (including `improgram`, `imquotacheck`, and `mboxutil`) hard-code use of the 'uid' attribute and may not work correctly with an alternate attribute name.

Although this option is documented to permit a list of LDAP attribute names, that facility did not work prior to Messaging Server 8.0.2 and as of MS 8.0.2, only the first attribute name in the list will be honored for certain operations (including MTA and store authentication).

Regarding the use of the LDAP attribute named by `ldap_uid`, normally `uid`, and its valid values: As of MS 6.2, the MTA checks that there is only one such attribute; as of 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 `uid`s.) 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.

52.15.6.10 Direct LDAP attribute name MTA options: `ldap_mlsrange` (LDAP attribute name)

RESTRICTED: Not yet used.

52.15.6.11 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.

52.15.6.12 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 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 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.

52.15.6.13 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 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 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.

52.15.6.14 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 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 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.

(Note also that the `acceptalladdresses` channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.15.6.15 Direct LDAP attribute name MTA options: `ldap_source_channel` (LDAP attribute name)

(New in 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.

52.15.6.16 Direct LDAP attribute name MTA options: ldap_source_optin*N* for *N*=1--8 (LDAP attribute name)

(New in MS 6.3-0.15) The ldap_source_optin*N* 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_optin*N* and ldap_optin*N* could be set to the same LDAP attribute.

The LDAP attributes named by ldap_source_optin*N* are the sending user analogues of ldap_optin*N*. In particular, the presence in a user entry of the LDAP attribute named by ldap_source_optin*N* normally (but see the spamfilter*N*_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.

52.15.6.17 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 MS 6.3-0.15, the MTA has the ability to chose 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 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.

52.15.6.18 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.

52.15.6.19 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.

52.15.6.20 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 MS 6.0) the usual mailHost and mailDeliveryOption interpretation.

52.15.6.21 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. In particular, note that the MTA (with normal delivery_options configuration) interprets absence of this attribute on a user entry as meaning to perform delivery to the user mailbox, and interprets absence of this attribute on a group entry as meaning to perform delivery to the group members. (This provides sensible and useful behavior for user or group entries lacking any explicit mailDeliveryOption value. However, note that this behavior also has implications when updating a user or group entry: in particular, if deleting the one and *only* mailDeliveryOption setting from a user or group, consider what delivery behavior you may wish to *explicitly set* in its place, as in the absence of any explicit setting you will get normal delivery; explicitly setting a value of nomail, or setting the user status or group status to a value such as inactive or disabled can be done if you truly desire *no* delivery effect.)

The schema sets an ACI on the default attribute, mailDeliveryOption, to allow user modification.

52.15.6.22 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.

52.15.6.23 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`.

52.15.6.24 Direct LDAP attribute name MTA options: `ldap_sender_sieve` (LDAP attribute name(s))

New in MS 8.0.1. The `ldap_sender_sieve` MTA option specifies the name of an LDAP attribute -- or a list of such names -- used to store a specific-to-that-ser Sieve filter that is applied to messages sent by that authenticated user.

This option has no default value.

52.15.6.25 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`.

52.15.6.26 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.

52.15.6.27 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).

52.15.6.28 Direct LDAP attribute name options: ldap_optin*N* (list of LDAP attribute names)

The ldap_optin*N*, *N*=1,...,8, MTA options may be used to name user or group LDAP attributes whose presence in an entry normally (but see the spamfilter*N*_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.

See also the ldap_source_optin*N* MTA options, which serve an analogous purpose for messages coming *from* a user or group.

These MTA options have no default, though the *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_optin`*`N`*; or for aliases defined in the alias file, the analogous setting is the Alias file named parameter [OPTIN*n*].

52.15.6.29 Direct LDAP attribute name options: `ldap_optout`*`N`* (list of LDAP attribute names)

The `ldap_optout`*`N`*, *`N`*=1,...,8, MTA options may be used to name user or group LDAP attributes whose presence in an entry normally (but see the `spamfilter`*`N`*`_null_optin` MTA options) causes messages addressed *to* that user or group to be "opted-out" of spam/virus filter package processing (by spam/virus filter package *`N`*).

Note that the scope of opt-outs only extends to the immediate expansion values of the LDAP entry - if the entry expands to another alias that alias will not honor the outer level opt-out.

These MTA options have no default.

For aliases defined via an alias option, the analogous option is `alias_optin`*`N`*; or for aliases defined in the alias file, the analogous setting is the Alias file named parameter [OPTOUT*n*].

52.15.6.30 Direct LDAP attribute name MTA options: `ldap_presence` (LDAP attribute name)

RESTRICTED: Not yet used.

52.15.6.31 Direct LDAP attribute name MTA options: `ldap_autosecretary` (LDAP attribute name)

RESTRICTED: Not yet used.

52.15.6.32 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`.

52.15.6.33 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. Futhermore, 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 this attribute and inclusive limit processing is in effect, or before the specified limit if exclusive time limit processing is in effect. No start date limit is enforced if this attribute is missing.

The schema sets an ACI on the default attribute, `vacationStartDate`, to allow user modification.

52.15.6.34 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/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. Futhermore, 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 before the time specified by this attribute and inclusive limit processing is in effect, or after the specified limit if exclusive time limit processing is in effect.

The schema sets an ACI on the default attribute, `vacationEndDate`, to allow user modification.

52.15.6.35 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`.

52.15.6.36 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`.

52.15.6.37 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 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`.

(Note that the `acceptalladdresses` channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.15.6.38 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.

52.15.6.39 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.

52.15.6.40 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.

52.15.6.41 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.

52.15.6.42 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.)

52.15.6.43 Direct LDAP attribute name MTA options: ldap_spare_*N* (LDAP attribute name)

The ldap_spare_*N*, *N*=1,...,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,...,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 $*N*E substitution sequence.

For *N*=1,...,6, the value of the named attribute may optionally (see the include_spares1 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_spares2 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 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.

52.15.6.44 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/mailing 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/mailing list entries. See the delivery_options MTA option for some discussion regarding enabling use of this attribute on group/mailing 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.

52.15.6.45 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/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. Futhermore, 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 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 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.

New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form `$<attribute><part>` where `$<attribute>` is "B" for the start (beginning) date or "E" for the end date, and `<part>` is one of the date parts defined in RFC 5260 section 4.2. So for example, the string `$EDATE"` would substitute in the end date in YYYY-MM-DD format. These substitution strings are treated as regular text if the corresponding attribute is not defined or is set to an invalid value.

Note that new in 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.

52.15.6.46 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. Futhermore, 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 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.

New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form `$<attribute><part>` where `$<attribute>` is "B" for the start (beginning) date or "E" for the end date, and `<part>` is one of the date parts defined in RFC 5260 section 4.2. So for example, the string `$EDATE"` would substitute in the end date in YYYY-MM-DD format. These substitution strings are treated as regular text if the corresponding attribute is not defined or is set to an invalid value.

Note that new in 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.)

52.15.6.47 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/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. Futhermore, 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 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 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.

New in MS 8.0.2.2, substitutions from the vacationStartDate and vacationEndDate are also also available. These substitutions take the form `$<attribute><part>` where `$<attribute>` is "B" for the start (beginning) date or "E" for the end date, and `<part>` is one of the date parts defined in RFC 5260 section 4.2. So for example, the string `$EDATE"` would substitute in the end date in YYYY-MM-DD format. These substitution strings are treated as regular text if the corresponding attribute is not defined or is set to an invalid value.

Note that new in 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.)

52.15.6.48 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/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. Futhermore, 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/mailing 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 MS 6.3-0.15, the MTA supports use of language-tagged values for the attribute named by ldap_autoreply_addresses.

52.15.6.49 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/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. Futhermore, 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 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.

52.15.6.50 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.

52.15.6.51 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.

52.15.6.52 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`.)

52.15.6.53 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.

52.15.6.54 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 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.

52.15.6.55 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.

52.15.6.56 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.

52.15.6.57 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.

52.15.6.58 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.

52.15.6.59 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.)

52.15.6.60 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).

52.15.6.61 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.

52.15.6.62 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 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.

52.15.6.63 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.

52.15.6.64 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 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.

52.15.6.65 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.)

(Note that the acceptalladdresses channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.15.6.66 Direct LDAP attribute name MTA options: ldap_maximum_messages_per_day (LDAP attribute name)

RESTRICTED: Not yet fully implemented.

52.15.6.67 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 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.

52.15.6.68 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 DNs 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 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.

52.15.6.69 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>`".)

52.15.6.70 Direct LDAP attribute name MTA options: `ldap_group_url1` (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`.

52.15.6.71 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`.

52.15.6.72 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.

52.15.6.73 ldap_group_dn2 Option

The ldap_group_dn2 MTA option names an LDAP attribute in which to store a list of DNs, *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.

52.15.6.74 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.

52.15.6.75 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 resulted 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.

52.15.6.76 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 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 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 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.

52.15.6.77 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*.

52.15.6.78 Direct LDAP attribute name MTA options: `ldap_digest_interval` (LDAP attribute name)

RESTRICTED. Not yet fully implemented.

52.15.6.79 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 messagtes 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.

52.15.6.80 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.

52.15.6.81 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 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).

52.15.6.82 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.

Support for HTML prefix text, insertion of text into text/html parts, and processing multiple parts in multipart/alternative was added in MS 8.0. As of MS 8.0.1.3 control over the HTML attribuates on the inserted text is also possible; see the prefix_text_attr MTA option for details.

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.

52.15.6.83 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.

Support for HTML suffix text, insertion of text into text/html parts, and processing multiple parts in multipart/alternative was added in MS 8.0. As of MS 8.0.1.3 control over the HTML attributes on the inserted text is also possible; see the `suffix_text_attr` MTA option for details.

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 `addsuffix` Sieve action.

52.15.6.84 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)!

52.15.6.85 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.

52.15.6.86 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.

52.15.6.87 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.

52.15.6.88 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.

52.15.6.89 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.

52.15.6.90 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.

52.15.6.91 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.

52.15.6.92 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).

52.15.6.93 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.

52.15.6.94 `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`.

52.15.6.95 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.

52.15.6.96 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.

52.15.6.97 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.

52.15.6.98 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.

52.15.6.99 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.

52.15.6.100 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.

52.15.6.101 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 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.

(Note that the acceptalladdresses channel option, if used, modifies the timing and form of the rejection.)

52.15.6.102 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 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 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 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.

(Note that the acceptalladdresses channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.15.6.103 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.

52.15.6.104 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.

52.15.6.105 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 MS 6.2, N=5--8 new in 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.

52.15.6.106 Direct LDAP attribute name MTA options: ldap_domain_attr_presence (LDAP attribute name)

RESTRICTED. Not yet used.

52.15.6.107 Direct LDAP attribute name MTA options: ldap_domain_attr_autosecretary (LDAP attribute name)

RESTRICTED. Not yet used.

52.15.6.108 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.

52.15.6.109 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.

52.15.6.110 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).

52.15.6.111 Direct LDAP attribute name MTA options: `ldap_domain_attr_disk_quota` (LDAP attribute name(s))

RESTRICTED. Not yet fully implemented.

52.15.6.112 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`.

52.15.6.113 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.

52.15.6.114 Direct LDAP attribute name MTA options: `ldap_domain_attr_sender_sieve` (LDAP attribute name(s))

New in MS 8.0.1. The `ldap_domain_attr_sender_sieve` MTA option specifies the name of an LDAP attribute (by default `mailDomainSenderSieve`) -- or a list of such names -- used to store a specific-to-that-domain Sieve filter that is applied to messages sent by authenticated users associated with that domain.

52.15.6.115 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.

52.15.6.116 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.

52.15.6.117 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*).

52.15.6.118 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. (The other output flags supported by the FORWARD mapping table are not supported by a domain catchall mapping.)

Flag comparisons supported by the FORWARD mapping table are, in general, also supported in a domain catchall mapping. This includes more recently added flag comparisons, including: 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. New in MS 8.0, support for $:R or $;R, and $:U or $;U tests. (However, the FORWARD mapping table's $:V or $;V flag test is *not* supported in a domain catchall mapping.)

52.15.6.119 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 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 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.

(Note that the acceptalladdresses channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.15.6.120 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.

52.15.6.121 Direct LDAP attribute name MTA options: `ldap_domain_attr_prefix_text` (LDAP attribute name(s))

The `ldap_domain_attr_prefix_text` MTA option specifies the name of a domain LDAP attribute which in turn is used to specify text to insert at the top of messages. The default LDAP attribute used for this purpose if this option is not specified is `mailDomainPrefixText`.

Any text specified by the attribute will be inserted into messages submitted by any authenticated user associated with the domain. Either plain text or HTML can be used. A maximum of 4096 octets from the attribute value will be processed.

HTML values are indicated in a prefix addition by enclosing the addition in <html></html> tags. (These tags are removed from the addition.)

Prefix additions to text/html parts are placed immediately after the <body> tag.

Plain text additions to text/html are preceded by
<pre> and followed by </pre>
. HTML additions to text/plain parts have all HTML tags removed and all entities are converted to corresponding characters. Character set conversions are performed as needed; the attribute value must be specified in UTF-8.

52.15.6.122 Direct LDAP attribute name MTA options: `ldap_domain_attr_suffix_text` (LDAP attribute name(s))

The `ldap_domain_attr_suffix_text` MTA option specifies the name of a domain LDAP attribute which in turn is used to specify text to insert at the bottom of messages. The default LDAP attribute used for this purpose if this option is not specified is `mailDomainSuffixText`.

Any text specified by the attribute will be inserted into messages submitted by any authenticated user associated with the domain. Either plain text or HTML can be used. A maximum of 4096 octets from the attribute value will be processed.

HTML values are indicated in a suffix addition by enclosing the addition in <html></html> tags. (These tags are removed from the addition.)

Suffix additions to text/html parts are placed immediately before the </body> tag.

Plain text additions to text/html are preceded by
<pre> and followed by </pre>
. HTML additions to text/plain parts have all HTML tags removed and all entities are converted to corresponding characters. Character set conversions are performed as needed; the attribute value must be specified in UTF-8.

52.15.6.123 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 identifical, effect.

New behavior in 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 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.

52.15.6.124 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 identifical, effect.

New behavior in 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 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.

52.15.6.125 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.

52.15.6.126 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).

52.15.6.127 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).

52.15.7 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.

52.15.7.1 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`.

52.15.7.2 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.

52.15.7.3 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 `saslswitchchannel` channel option is set). The default LDAP attribute name, if this option is not set, is `mailSMTPSubmitChannel`.

52.15.7.4 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.

52.15.7.5 LDAP attributes returned upon authentication MTA options: `ldap_auth_attr_hold_for` (LDAP attribute name)

(New in MS 8.0)

52.15.8 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 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.

52.15.8.1 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.

52.15.8.2 Domain match cache control (`domain_match_cache_size`, `domain_match_cache_timeout`)

As of 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 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.

52.15.8.3 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.

52.15.8.4 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.

52.15.8.5 URL result case sensitivity option (`url_result_cache_case`)

New in MS 8.1.0.2. The `url_result_cache_case` MTA option controls whether or not the URL result cache entry names are case sensitive. Setting the option to 1 makes them case sensitive. The default is 0, in which case the entry names are case-insensitive.

This option needs to be set to 1 if case-sensitive LDAP attributes are being used. Note that this may have an adverse effect on cache efficiency.

52.15.8.6 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.

52.16 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. But `tmpdir` (replacing the former `imta_tmp` MTA Tailor option) and `langdir` (replacing the former `imta_lang` MTA Tailor option) are still configurable; (note that in legacy configuration, these two options are MTA option file options).

52.16.1 Directory location MTA options: `tmpdir` (directory path)

The `tmpdir` MTA option (prior to MS 7.0, this MTA option was instead called `imta_tmp` and located in the MTA Tailor file, whereas `tmpdir` in legacy configuration is an MTA option file `option.dat` option) specifies the temporary file directory; it defaults to `/tmp/` (trailing slash included).

On Linux, this option should instead be set to `/dev/shm/`.

52.16.2 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. (Prior to MS 7.0, this was called `imta_lang`, and located in the MTA Tailor file.)

52.17 DKIM MTA options

New in Messaging Server 7.0.5 are several MTA and channel options that affect message handing based on the presence of certain DKIM signatures. See also the `dkim*` and `destinationdkim*` channel options, which perform a similar function on a per-channel basis.

52.17.1 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.

52.17.2 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.

52.17.3 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.

52.18 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.

52.18.1 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]
```

52.18.2 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 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 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.

52.19 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.

52.19.1 Error text and error interpretation MTA options: `access_errors` (0 or 1)

As of 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 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 MS 6.2, this (control of the error text due to group id mismatch) was the only purpose and only effect of this option.

52.19.2 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 CRLF, and not including the leading numeric error code and extended error code).

Table 52.16 `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 `internet.rules` file would result in this error. As of Messaging Server 7.0.5, the `internet.rules` file was modified to consist of solely a "`.`" rule which consults the set of top level domain names from the `tlds.txt` file -- so in Messaging Server 7.0.5 or later, an out-of-date `tlds.txt` file or an attempt to submit a message to a top-level Internet domain not specified in the `tlds.txt` 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 uid found during an `alias_urlW` 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 `viaaliasrequired` 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 `mailRoutingSmartHost`.
`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 `access_errors` 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 `*_ACCESS` mapping table 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 `[AUTH_LIST]` 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 `ldap_reject_text` MTA option and `alias_error_text` 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 `error_text_alias_fileexist`).
`error_text_alias_fileexist`	452 or 550+	4.5.0	nonexistant 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 `mailHost` 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 blocklimit channel keyword setting *and* 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 sourceblocklimit option setting, or the block_limit 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 sourceblocklimit or block_limit 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 error_text_message_too_large 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 linelimit 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 *not* 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 line_limit MTA option setting. Thus normally the following non-configurable text is used: "a message *x* lines long exceeds the line limit of *y* 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 block_size MTA option), configured via the [BLOCKLIMIT] named mailing list parameter for a list defined in the alias file or alias database, or via the alias_blocklimit alias option (Unified Configuration), or configured via the user/group-level mgrpMsgMaxSize attribute (more precisely, the attribute named by the ldap_maximum_message_size MTA option) or domain-level mailDomainMsgMaxBlocks

				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 `mailMsgMaxBlocks` attribute (more precisely, the attribute named by the `ldap_blocklimit` MTA option) or domain-level `mailDomainMsgMaxBlocks` 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 $S 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	This error is returned when the MTA detects that a user's personal status (inetUserStatus or mailUserStatus) or the user's domain status (mailDomainStatus) is disabled during alias expansion.
error_text_disabled_alias	550 or 450*	5.2.1	alias disabled; cannot receive new mail	
error_text_over_quota	451 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_lmtpc*) 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_illegal_8bit_from	553	5.1.3	illegal 8bit characters in return address	Incorrect eight bit data present in the return address.
error_text_disallowed_8bit	553	5.1.0	8bit characters in address not allowed in this context	Eight bit data present in an address when eight bit is not allowed.
error_text_disallowed_8bit_from	553	5.1.0	8bit characters in return address not allowed in this context	Eight bit data present in the return address when eight bit is not allowed.
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 processing") 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	This error is returned when the MTA detects that a group's status (the value of the inetMailGroupStatus attribute, or more precisely, the values of whatever LDAP attributes are named by the ldap_group_status or ldap_group_mail_status MTA options) or the group's domain status (mailDomainStatus, or more precisely the attribute named by the ldap_domain_attr_mail_status MTA option) is inactive during alias expansion.
error_text_disabled_group	550	5.2.1	group disabled; cannot receive new mail	This error is returned when the MTA detects that a group's status (the value of the inetMailGroupStatus attribute, or more precisely, the values of whatever LDAP attributes are named by the ldap_group_status or ldap_group_mail_status MTA options) or the group's domain status (mailDomainStatus, or more precisely the attribute named by the ldap_domain_attr_mail_status MTA option) is disabled during alias expansion.
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 mistake was made while provisioning users and domains in the directory.
error_text_spamfilter_error	451	4.7.1	filtering/scanning error	As of MS 6.3, a synonym for the new-in-6.3 error_text_spamfilter1_error MTA

				option; as of MS 6.3, obsolete and used only if `error_text_spamfilter1_error` is not set.
`error_text_spamfilter`*N*`_error`	451	4.7.1	filtering/scannning error	New in MS 6.3. The default error text to use when there is a problem attempting to use the *N*th 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. As of 7.0.5.37, any value specified for this option unconditionally overrides any error text returned by the filter package.
`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 MS 6.1.
`error_text_nosolicit`	550 or 530++	5.7.1	solicitations of this type are not allowed	New in 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	The error text returned when the MTA's attempt to SRS/MUL decode an address encounters a syntax error in the SRS/MUL encoding.
`error_text_srs_timeout`	550	5.7.1	SRS/MUL address has timed out	Error text returned when the MTA attempts to decode an SRS/MUL encoded address whose timestamp has expired.
`error_text_srs_badhash`	550	5.7.1	SRS/MUL address has a bad hash value	Error text returned when the MTA's attempt to decode an SRS/MUL encoded address finds an invalid hash value.
`error_text_spf_temperror_4`	451	4.7.24	temporary error in SPF verification of MAIL FROM domain	(New in MS 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 MS 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 MS 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 MS 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 MS 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 MS 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 (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. 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`
error_text_spf_softfail_4	451	4.7.23	SPF verification of MAIL FROM domain soft failed (*domain*)	(New in MS 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 (spfhelo) time, the error_text_spf_ehlo_softfail_4 text is used instead.
error_text_spf_softfail_5	550	4.7.23	SPF verification of MAIL FROM domain soft failed	(New in MS 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 (`spfhelo`) time, the `error_text_spf_ehlo_softfail_5` text is used instead.
`error_text_spf_ehlo_temperror_4`	451	4.7.24	temporary error in SPF verification of EHLO/HELO domain	(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.
`error_text_spf_helo_temperror_5`	550	4.7.24	temporary error in SPF verification of EHLO/HELO domain	(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.
`error_text_spf_helo_permerror_4`	451	5.7.24	permanent error in SPF verification of EHLO/HELO domain	(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
`error_text_spf_ehlo_permerror_5`	550	5.7.24	permanent error in SPF verification of EHLO/HELO domain	(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 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	(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. 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`
`error_text_spf_ehlo_fail_5`	550	5.7.23	SPF verification of EHLO/HELO domain failed	(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. 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`
`error_text_spf_ehlo_softfail_4`	451	4.7.23	SPF verification of EHLO/HELO domain soft failed	(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 `spfhelo`), 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.
error_text_spf_ehlo_softfail_5	550	4.7.23	SPF verification of MAIL FROM domain soft failed	(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.
error_text_mailfromdnsverify	550 or 450	5.1.8 or 4.1.8	invalid/host-not-in-DNS return address not allowed	(New in 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.
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 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 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 MS 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 MS 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 MS 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 MS 6.3) Currently unused.
error_text_transaction_limit_exceeded	450	4.5.3	number of transactions exceeds allowed maximum	(New in MS 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 MS 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 MS 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 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 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 MS 6.2.
error_text_spare1_error				(New in 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 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 can be provided in the message-id field of MTA connection transaction logging if the MTA option log_message_id is enabled.

Table 52.17 MTA AUTH errors

SASL error or code	SMTP code	Extended code	Basic SMTP error text+	log_message_id text in "U" record	Notes
HULA_BADPARAM or HULA_NOMEM	450	4.3.0	SASL initialization failed; server unavailable		LDAP server unavailable/unresponsive for authentication; proxy authentication not properly configured or trouble performing it; server running out of memory; *etc.*

SASL_UNAVAIL	454	4.7.0	Authentication server unavailable	Authentication server unavailable	LDAP server unavailable/unresponsive for authentication.
	503	5.7.0	AUTH command already issued		SMTP AUTH already successfully performed.
	533	5.7.1	AUTH command is not enabled		No maysaslserver or mustsaslserver enabled on channel.
	501	5.7.0	Cannot decode BASE64	Cannot decode BASE64	Argument right of = fails to BASE64 decode; or, argument on new line fails to BASE64 decode.
	503	5.7.1	Mail transaction already in progress		SMTP AUTH not permitted now that message submission has begun.
	501	5.7.0	AUTH operation aborted by client	Client aborted AUTH operation	
SASL_OK	235	2.7.0	*mechanism* 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_PWLOCK	534	5.7.8	Bad username or password	Password/account is locked	(New in MS 8.0.2)
SASL_WEAKPASS	534	5.7.9	Password is too weak	Password is too weak	(New in MS 8.0)
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	538	5.7.11	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	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 authentication error	Unknown AUTH errors <sasl-errno> <sasl-aux-errno>	

+ 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_message_id field was the empty string for this success case

52.19.3 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 52.18 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.

Prior to the 8.0.1.2 release the MTA converts the "defer" mailUserStatus value to "inactive" internally when performing a reprocessing operation. This means that use of the rehostuser utility, which makes use of "defer", in conjunction with setting bit 0 (value 1) of use_permanent_error can cause messages received during a rehostuser operation to be returned as undeliverable.

This issue has been addressed in 8.0.1.2 and later by having the MTA convert "defer" status to "hold" status instead.

IMPORTANT NOTE: More generally, various parts of Messaging Server, other Communications Suite components, and even external provisioning utilities assume that "inactive" status is used to indicate transient rather than permanent problems. As such, setting bit 0 (value 1) of use_permanent_error to cause a permanent error status to be returned is NOT RECOMMENDED.

52.19.4 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 52.19 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.

52.20 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.

52.20.1 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.

52.20.2 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.

52.20.3 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".

52.21 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.

52.21.1 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*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).

52.21.2 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.

52.21.3 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.

52.21.4 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 "#".

52.21.5 Debug MTA options: debug_flush (0 or 1)

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 Dispatcher debug output.

52.21.6 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**.

52.21.7 File format MTA options: fdirectory (0 or 1; OpenVMS only)

The fdirectory option is only available on OpenVMS.

52.21.8 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).

52.21.9 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.

52.21.10 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.

52.21.11 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 MTA Tailor file option, in legacy configuration). (Prior to MS 6.0, the area such files were written to was controlled by the now obsolete imta_scratch MTA 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 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 (in the tmpdir or imta_tmp directory). Prior to this change, the LMTP server did not do in-memory buffering and instead always buffered to temporary files (in imta_tmp).

52.21.12 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.

52.21.13 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.

52.21.14 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.

52.21.15 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.

52.21.16 projectid Option Under mta

The projectid 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.

52.21.17 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.

52.21.18 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).

52.21.19 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. As of MS 8.0.2.3, the bits in the option default to 1 if the corresponding URL option is not set and 0 otherwise. The default value was 8 prior to MS 8.0.2.3. The default is 0 in versions of Messaging Server prior to 8.0.2.3. 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 MS 6.3P1, 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).

52.22 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 accomodate 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 on-going quest to improve performance and scalability may mean that these documented values become out-of-date.

52.22.1 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 explicited configured for this option) will result in an mm_init error, "no room in table for alias ...".

52.22.2 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 ...".

52.22.3 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 ...".

52.22.4 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.

52.22.5 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).

52.22.6 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).

52.22.7 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.

52.22.8 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 ...".

52.22.9 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 ...".

52.22.10 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 ...".

52.22.11 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.

52.22.12 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 MS 6.2 to 2,000,000 for 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 ...".

52.22.13 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.

52.22.14 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.".

52.22.15 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.".

52.22.16 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.

52.22.17 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".

52.22.18 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.

52.22.19 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 ...".

52.22.20 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 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 MS 6.3; (10,000,000 in 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 MS 6.3) will result in the maximum value (of 10,000,000, or 50,000,000 as of 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.

52.22.21 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.

52.23 Latency server MTA options

New in the 8.1.0.6 release, as part of the smartsend facility the MTA supports sending information about message latency to a separate server in JSON format. latency_* MTA options control the MTA's connections to this server.

At present there are two fields in each JSON object: "ts" is a timestamp expressed in in milliseconds since the UNIX epoch, and "la" is the message processing latency in milliseconds. For example:

```
{"ts":1589476358903,"la":95109}
```

52.23.1 Latency server options: latency_host (host)

New in MS 8.1.0.6. The latency_host MTA option specifies the host name (or IP address) of the host providing a server that accepts message latency information from the MTA. The default is 127.0.0.1 since the usual configuration is to have a co-resident latency server.

52.23.2 Latency server options: latency_port (port)

New in MS 8.1.0.6. The latency_port MTA option specifies the port number for the server that accepts message latency information from the MTA. If not specified, it defaults to 7072, the usual port for latency servers.

52.23.3 Latency server options: latency_expire (integer)

New in MS 8.1.0.6. The latency_expire MTA option specifies the amount of time, in seconds, that a connection to a latency server can remain idle and still be eligible for reuse.

52.23.4 Latency server options: latency_timeout (integer)

New in MS 8.1.0.6. The latency_timeout MTA option specifies the amount of time, in seconds, to wait on a connection to a latency server.

52.23.5 Latency server options: latency_max_failures (integer)

New in MS 8.1.0.6. The latency_max_failures MTA option specifies the number of attempts the MTA will make to connect to the latency server before disabling latency server support. Note that this is a per-process limit and only counts failures attempts 5 or more seconds apart.

52.24 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.

52.24.1 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.

52.24.2 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.

52.24.3 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.

52.24.4 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).

52.24.5 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.

52.25 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: (targer="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";)
```

52.25.1 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).

52.25.2 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.

52.25.3 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).

52.25.4 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 ldapportPAB option (local.service.pab.ldapport configutil parameter in legacy configuration).

52.25.5 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).

52.26 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.

52.26.1 Alias and address MTA options: alternate_recipient (string)

(New in MS 8.0.1.) The alternate_recipient MTA option specifies the comment string, including the surrounding parentheses, that is used to specify an alternate recipient address as part of a mailing list address entry. The default value for this string is (ALTERNATE-RECIPIENT).

For example, assuming the default value of this option, an entry of the form:

```
listmember@domain.com (alternate-recipient listalternate@domain.com)
```

would associated the alternate address listalternate@domain.com with the mailing list address listmember@domain.com.

52.26.2 alternate_recipient_mode Option

The alternate_recipient_mode MTA option controls the order in which additional alternate recipients are added to an existing alternate recipient list. Possible values are:

0 old recipients follow new recipient

1 new recipient follows old recipient

2 new recipient replaces any old recipients

3 new recipient is silently dropped if any old recipients are present

The default is 0, which is consistent with military messaging requirements.

52.26.3 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.

52.26.4 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.

52.26.5 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. See also the expn* channel options.

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.

52.26.6 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).

52.26.7 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 for 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).

52.26.8 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).

52.27 MAILSERV MTA options

RESTRICTED. MAILSERV is not yet fully implemented.

52.27.1 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.

52.27.1.1 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.

52.27.1.2 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.

52.27.1.3 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.

52.27.2 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.

52.27.3 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.

52.27.4 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.

52.27.5 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 mgrpListName LDAP attribute is really a use of the attribute named by the ldap_list_name MTA option.

52.28 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.

52.28.1 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.

52.28.1.1 Access mapping table MTA options: access_auth (bitmask)

(New in MS 8.0) 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 MAIL FROM command's 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.

52.28.1.2 Access mapping table MTA options: access_counts (bitmask)

(New in 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/expansion-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).

New in 8.0.1.2 is *expansion-count*, a count of the number of actual recipient addresses produced by the current RCPT TO command, including the current one.

Table 52.20 access_counts MTA option bit values

Bit	Value	Usage
0	1	If set, include recipient counts in each [ORIG_]SEND_ACCESS and [ORIG_]MAIL_ACCESS mapping table probe
1	2	If set, include recipient counts in each ORIG_SEND_ACCESS probe
2	4	If set, include recipient counts in each SEND_ACCESS probe
3	8	If set, include recipient counts in each ORIG_MAIL_ACCESS probe
4	16	If set, include recipient counts in each MAIL_ACCESS probe

Bit 0 is the least significant bit.

52.28.1.3 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 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 52.21 `access_orcpt` MTA option bit values

Bit	Value	Usage
0	1	If set, include ORCPT value in each `[ORIG_]SEND_ACCESS` and `[ORIG_]MAIL_ACCESS` mapping table probe
1	2	(New in 6.3) If set, include ORCPT value in each `ORIG_SEND_ACCESS` probe
2	4	(New in 6.3) If set, include ORCPT value in each `SEND_ACCESS` probe
3	8	(New in 6.3) If set, include ORCPT value in each `ORIG_MAIL_ACCESS` probe
4	16	(New in 6.3) If set, include ORCPT value in each `MAIL_ACCESS` probe

Bit 0 is the least significant bit.

52.28.1.4 Access mapping table MTA options: `include_connectioninfo` (bitmask)

(New in 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 52.22 `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.

52.28.1.5 Access mapping table MTA options: `include_conversiontag` (bitmask)

New in 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 52.23 `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 after any authenticated sender and to any `ldap_spare_*` fields (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 `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)
6	64	MAIL_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)
7	128	FROM_ACCESS : (New in 7.0-3.01? post MS 6.3) 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)
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, `|`, 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 *_ACCESS 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 `include_conversiontag`.
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, `|`, 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, `|`, delimiter).
11	2048	AUTH_ACCESS mapping: (New in 8.0.2.2) include as a field after the username, but before the destination system in the probe (with a vertical bar, `|`, delimiter).

12	4096	DEQUEUE_ACCESS mapping: (New in 8.0.2.3) include as a field after the auth-sender, but before the priority in the probe (with a vertical bar, `|`, delimiter).
13	8192	AUTH_REWRITE mapping: (New in 8.1.0.1) include as a field after the auth-param, but before any extra header fields in the probe (with a vertical bar, `|`, delimiter).
14	16384	(New in 8.1.0.1) Include the global conversion tag list as an X-Tags: field in the first section of the second part of any DSNs that are generated. Note that this may expose internal information and is only intended for use when all DSNs are processed internally.
15	32768	(New in 8.1.0.1) Include the global conversion tag list in any SMTP_ACTIONS mapping probe.
16	65536	(New in 8.1.0.1) Include the global conversion tag list in any MX_ACCESS mapping probe.

Bit 0 is the least significant bit.

The default is 0, meaning that conversion tag values are not included in any mapping table probes.

52.28.1.6 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 includeed 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 52.24 `include_mtpriority` MTA option values

Bit	Value	Meaning
0	1	Include MT-PRIORITY and size estimate in `FROM_ACCESS` probes immediately after any `include_spares1` 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)
2	4	Include MT-PRIORITY and message size estimate in `ORIG_SEND_ACCESS` probes immediately after any `include_spares1` values
3	8	Include MT-PRIORITY and message size estimate in `SEND_ACCESS` probes immediately after any `include_spares1` values
4	16	Include MT-PRIORITY and message size estimate in `ORIG_MAIL_ACCESS` probes immediately after any `include_spares1` values
5	32	Include MT-PRIORITY and message size estimate in `MAIL_ACCESS` probes immediately after any `include_spares1` 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 includes 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.

52.28.1.7 Access mapping table MTA options: `include_retries` (bitmask)

New in MS 8.0.2.3. This option selectively enables the inclusion of message retry count information in various mapping table probes. When enabled, the number of previous retry attempts as well as a "skip count" will be included as a field in the probe, separated by a comma. A -1 will appear if the retry count or skip count, respectively, cannot be determined.

The skip count is an integer value encoded as part of the queue entry filename that is intended for use in keeping track of round-robin positioning in IP address lists.

FORWARD COMPATIBILITY NOTE: Additional comma-separated values may be added to this field in the future. Implementations should be aware of this and take appropriate steps to handle the field valye properly.

This option takes a bit-encoded integer value. The currently defined bits are shown in the following table:

Table 52.25 `include_retries` MTA option bit values

Bit	Value	Usage
0	1	AUTH_ACCESS : include after the filename and before the queue time
1	2	DEQUEUE_ACCESS : include after the filename and before the queue time
2	4	MESSAGE-SAVE-COPY : include after the filename

52.28.1.8 Access mapping table MTA options: `include_spares1` (bitmask)

(New in Messaging Server 7u2, renamed from `include_spares` to `include_spares1` in MS 8.0.2.2.) 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_spares1` MTA option. This option is bit-encoded, with the bits defined as follows:

Table 52.26 `include_spares1` MTA option values

Bit	Value	Meaning
0	1	Include sender `ldap_spare_1` attribute in FROM_ACCESS probes
1	2	Include sender `ldap_spare_2` attribute in FROM_ACCESS probes

2	4	Include sender ldap_spare_3 attribute in FROM_ACCESS probes
3	8	Include sender ldap_spare_4 attribute in FROM_ACCESS probes
4	16	Include sender ldap_spare_5 attribute in FROM_ACCESS probes
5	32	Include sender ldap_spare_6 attribute in FROM_ACCESS probes
6	64	Include recipient ldap_spare_1 attribute in ORIG_SEND_ACCESS probes
7	128	Include recipient ldap_spare_2 attribute in ORIG_SEND_ACCESS probes
8	256	Include recipient ldap_spare_3 attribute in ORIG_SEND_ACCESS probes
9	512	Include recipient ldap_spare_4 attribute in ORIG_SEND_ACCESS probes
10	1024	Include recipient ldap_spare_5" attribute in ORIG_SEND_ACCESS probes
11	2048	Include recipient ldap_spare_6 attribute in ORIG_SEND_ACCESS probes
12	4096	Include recipient ldap_spare_1 attribute in SEND_ACCESS probes
13	8192	Include recipient ldap_spare_2 attribute in SEND_ACCESS probes
14	16384	Include recipient ldap_spare_3 attribute in SEND_ACCESS probes
15	32768	Include recipient ldap_spare_4 attribute in SEND_ACCESS probes
16	65536	Include recipient ldap_spare_5 attribute in SEND_ACCESS probes
17	131072	Include recipient ldap_spare_6 attribute in SEND_ACCESS probes
18	262144	Include recipient ldap_spare_1 attribute in ORIG_MAIL_ACCESS probes
19	524288	Include recipient ldap_spare_2 attribute in ORIG_MAIL_ACCESS probes
20	1048576	Include recipient ldap_spare_3 attribute in ORIG_MAIL_ACCESS probes
21	2097152	Include recipient ldap_spare_4 attribute in ORIG_MAIL_ACCESS probes
22	4194304	Include recipient ldap_spare_5 attribute in ORIG_MAIL_ACCESS probes
23	8388608	Include recipient ldap_spare_6 attribute in ORIG_MAIL_ACCESS probes
24	16777216	Include recipient ldap_spare_1 attribute in MAIL_ACCESS probes
25	33554432	Include recipient ldap_spare_2 attribute in MAIL_ACCESS probes
26	67108864	Include recipient ldap_spare_3 attribute in MAIL_ACCESS probes
27	134217728	Include recipient ldap_spare_4 attribute in MAIL_ACCESS probes
28	268435456	Include recipient ldap_spare_5 attribute in MAIL_ACCESS probes
29	536870912	Include recipient ldap_spare_6 attribute in MAIL_ACCESS 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.

52.28.1.9 Miscellaneous mapping table MTA options: include_spares (bitmask)

Replaced in 8.0.2.2 by the include_spares1 MTA option.

52.28.1.10 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.

52.28.1.11 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.

52.28.1.12 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 value. 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 52.27 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 `ldap_auth_domain` LDAP attribute (*e.g.*, `mgrpAllowedDomain`) checks
13	8192	Mailing list `ldap_cant_domain` LDAP attribute (*e.g.*, `mgrpDisallowedDomain`) checks
14	16384	Mailing list `ldap_auth_url` LDAP attribute (*e.g.*, `mgrpAllowedBroadcaster`) checks
15	32768	Mailing list `ldap_cant_url` LDAP attribute (*e.g.*, `mgrpDisallowedBroadcaster`) 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 `ldap_moderator_url` LDAP attribute (*e.g.*, `mgrpModerator`) 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.

52.28.2 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. See also the `string_pool_size_1` option. 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.

52.28.2.1 `averages_cache_size` Option

Not currently implemented. The MeterMaid facility provides an alternate, more general, facility.

52.28.2.2 `averages_cache_timeout` Option

Not currently implemented. The MeterMaid facility provides an alternate, more general, facility.

52.28.2.3 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 52.28 include_spares2 MTA option values

Bit	Value	Meaning
0	1	Include sender ldap_spare_1 attribute in FORWARD probes
1	2	Include sender ldap_spare_2 attribute in FORWARD probes
2	4	Include sender ldap_spare_3 attribute in FORWARD probes
3	8	Include sender ldap_spare_4 attribute in FORWARD probes
4	16	Include sender ldap_spare_5 attribute in FORWARD probes
5	32	Include sender ldap_spare_6 attribute in FORWARD probes
6	64	(New in MS 8.0.2.2) Include recipient ldap_spare_1 attribute in FORWARD probes
7	128	(New in MS 8.0.2.2) Include recipient ldap_spare_2 attribute in FORWARD probes
8	256	(New in MS 8.0.2.2) Include recipient ldap_spare_3 attribute in FORWARD probes
9	512	(New in MS 8.0.2.2) Include recipient ldap_spare_4 attribute in FORWARD probes
10	1024	(New in MS 8.0.2.2) Include recipient ldap_spare_5 attribute in FORWARD probes
11	2048	(New in MS 8.0.2.2) Include recipient ldap_spare_6 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.

52.28.2.4 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.

52.28.2.5 Miscellaneous mapping table MTA options: `message_save_copy_flags` (bitmask)

(New in MS 6.3) 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 52.29 `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	8	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 (if the field is enabled).
4	16	(New in 8.0.1.3) Include the transaction log action code value for the first recipient in the mapping table probe, immediately after the MT-PRIORITY field (if the field is enabled). This single character will normally be a "D" when delivery was successful.
5	32	(New in 8.0.2.1) Include the destination host for the first recipient in the mapping table probe, immediately after the transaction action code value (if the field is enabled).

Bit 0 is the least significant bit.

The default is 0.

52.28.2.6 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.

52.28.2.7 Miscellaneous mapping table MTA options: use_comment_strings (bitmask)

(New in 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.

52.28.2.8 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 52.30 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, |) *source-channel*|*from-address*|*to-address* 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, |) *source-channel*|*from-address*|*to-address* 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.

52.28.2.9 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 cannot 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 52.31 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, |) *source-channel|destination-channel|address*
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, |) *source-channel|destination-channel|address*
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, |) *destination-channel|address*
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, |) *destination-channel|address*
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).)

Note that as of 8.0.1.3, the usereversedatabase source channel option can be used to override the setting of use_reverse_database on a channel by channel basis.

52.28.2.10 Miscellaneous mapping table MTA options: use_personal_names (bitmask)

(New in 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.

52.29 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.

52.29.1 Memcache MTA/channel options: memcache_host (host)

The memcache_host MTA/channel option specifies the host name (or IP address) of the host providing a server supporting the memcache protocol. (The server software is often, but not always, memcached.) There is no default.

Use of the memcache_host channel option is only supported in cases where a channel-specific memcache server makes sense semantically. Currently the only channels that support this usage are those associated with the MTA-STS server.

52.29.2 Memcache MTA/channel options: memcache_port (port)

The memcache_port MTA/channel option specifies the port number for the server providing support for the memcache protocol. If not specified, it defaults to 11211, the usual port for memcache servers.

Use of the memcache_port channel option is only supported in cases where a channel-specific memcache server makes sense semantically. Currently the only channels that support this usage are those associated with the MTA-STS server.

52.29.3 Memcache MTA options: memcache_expire (integer)

The memcache_expire MTA option specifies the amount of time, in seconds, that a connection to a memcache server can remain idle and still be eligible for reuse.

52.29.4 Memcache MTA options: memcache_timeout (integer)

The memcache_timeout MTA option specifies the amount of time, in seconds, to wait for a connection to a memcache server.

52.29.5 Hashing memcache keys: memcache_hash_algorithm (hash algorithm name)

New in MS 8.1.0.3. The memcache_hash_algorithm MTA option controls what hash algorithm the MTA uses to optionally hash memcache keys. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, SHA256, SHA512, MD128, or MD160. SHA1 is the default. Note that the setting of this option must be the same across a deployment for successful coordination of memcache access.

52.29.6 Memcache/Redis MTA options: alias_database_url (memcache:/redis: URL)

The value of the alias_database_url MTA option should be a either a memcache: URL or (as of MS 8.1) a redis: URL for storing alias database data.

52.29.7 Memcache/Redis MTA options: domain_database_url (memcache:/redis: URL)

The value of the domain_database_url MTA option should be either a memcache: URL or (as of MS 8.1) a redis: URL for storing domain database data (that is, supplementary rewrite rules).

52.29.8 Memcache/Redis MTA options: forward_database_url (memcache:/redis: URL)

The value of the forward_database_url MTA option should be either a memcache: or (as of MS 8.1) a redis:URL for storing forward database data. Note that forward_database_url 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.

52.29.9 Memcache/Redis MTA options: general_database_url (memcache:/redis: URL)

The value of the general_database_url MTA option should be either a fully specified memcache: or (as of MS 8.1) a fully specified redis:URL for storing general database data. For example, the URL "memcache://memcache.exmaple.com/" specifies that memcache protocol should be used to connect to memcache.exmaple.com on the default memcache port. Note that in the case of Redis the host to connect to is always given by the redis.hostlist option, so the URL is always "redis:///".

Note that the general_database_url option 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.

52.29.10 Memcache/Redis MTA options: reverse_database_url (memcache:/redis: URL)

The value of the reverse_database_url MTA option should be either a memcache: or (as of MS 8.1) a redis:URL for storing reverse database data. Note that reverse_database_url 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.

52.30 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.

52.30.1 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 52.32 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.

52.30.2 Direct LDAP MTA options: capture_domain_replace (0-2)

Normally the effects of domain-level capture LDAP attributes (as specified by the ldap_domain_attr_capture MTA option) and user-level capture LDAP attributies (as specified by the ldap_capture MTA option) are cumulative, that is, message copies are sent to all such capture/journal recipients.

The new-in 8.0.1.3 capture_domain_replace MTA option changes this behavior. If set to a positive value N, it will change the behavior of the first N attributes so that domain-level capture addresses are replaced rather than being combined with user-level attributes.

The default value of 0 disables replacement.

Note that the special attribute value "*" can be used as a "null" capture address. This makes it possible to disable domain-level capture on a per-user basis.

52.30.3 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, SHA256, SHA512, MD128, or MD160. The default is MD5.

52.30.4 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.

52.30.5 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.

52.31 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.

52.31.1 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.

Or for aliases stored outside of LDAP, the alias_blocklimit alias option (or [BLOCKLIMIT] alias file named parameter in legacy configuration) can be used to impose limits on a per-alias basis.

(Note also that the acceptalladdresses channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.31.2 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).

52.31.3 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).

52.31.4 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.

52.31.5 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 maxheaderaddrs and maxheaderchars channel options, and the MAXCHARS, MAXIMUM, and MAXLINES header trimming options.

52.31.6 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 that the linelimit channel option can be used to impose line limits on a per-channel basis. The alias_linelimit alias option (or [LINELIMIT] alias file named parameter in legacy configuration) may be used to impose line limits on a per-alias basis. Analogously, the block_limit MTA option may be used to set a limit based on number of blocks in a message.

(Note that the acceptalladdresses channel option, if used, may modify the timing and form of rejections due to exceeding message size constraints.)

52.31.7 Message size MTA options: local_quota_checks (integer)

RESTRICTED: The local_quota_checks MTA option is not fully implemented.

52.31.8 Message fragmentation size limit MTA options: max_header_block_use (real number strictlybetween 0 and 1), max_header_line_use (real number strictly between 0 and 1)

52.31.8.1 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.

52.31.8.2 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.

52.31.9 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.

52.31.10 Message size MTA options: max_header_lines (integer)

The max_header_lines MTA option causes truncatation 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.

52.31.11 Maximum message levels/parts to process MTA options: max_mime_levels (integer), max_mime_parts (integer)

52.31.11.1 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.

52.31.11.2 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.

52.31.12 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.

52.31.12.1 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.

52.31.12.2 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.

52.31.12.3 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.

52.31.12.4 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.

52.32 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.

52.32.1 Tracking hash function usage: tracking_hash_algorithm (hash algorithm name)

The tracking_hash_algorithm MTA option controls what hash algorithm the MTA uses to generate hashes of tracking and recall secrets for use with the MTQP protocol. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, SHA256, SHA512, MD128, or MD160. The default if this option is not specified is SHA1, as required by the MTQP protocol standard.

Important note: As of this writing (February 2020) there is no practical preimage attack on SHA-1, which is what would be required to attack its usage in MQTP. (All known practical attacks are collision attacks, which generate a collision for a known preimage.) As such, the only justification for switching to SHA-2 hash function is to meet compliance requirements, and the benefits of meeting such requirements in the absence of any technical justification must be carefully weighed against the costs of standards incompliance and being incompatible with other MTQP implementations.

52.32.2 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.

52.32.3 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.

52.33 MeterMaid MTA options

MTA options for setting general MeterMaid configuration parameters were introduced in Messaging Server 7.2-7.02; previously (from 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.

52.33.1 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.

52.33.2 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.

52.33.3 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.

52.33.4 MeterMaid MTA options: metermaid_port (port)

The metermaid_port MTA option specifies the MeterMaid port for the Sieve metermaid operator. This MTA option is 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.

52.33.5 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 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.

52.33.6 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.

52.34 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_mlsrange` 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.

52.34.1 `mls` Option

RESTRICTED. Not yet fully implemented.

52.35 MTQP MTA options

New in the 8.0 release is MTQP (Message Tracking and Query Protocol) support.

52.35.1 MTQP MTA options: `mtqp_port` (port)

New in MS 8.0.

52.35.2 MTQP MTA options: `mtqp_timeout` (integer)

New in MS 8.0.

52.35.3 MTQP MTA options: `mtqp_expire` (integer)

New in MS 8.0.

52.36 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.

52.36.1 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).

52.36.2 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.

52.36.3 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.

52.36.4 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).

52.36.5 Notification message MTA options: notary_decode (-1, 0, or 1)

When the MTA is generating a DSN or MDN and using a %H substitution sequence to insert original message headers into that DSN or MDN, 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 disposition_prefix.txt file being used, or forcibly converted to the return_prefix.txt character set or disposition_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 character set specified in the disposition_prefix.txt file is the character set used for the first, human readable portion of the MDN.)

The default value is 0, meaning that encoded-words that match the character set specified in the return_prefix.txt file or disposition_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 or disposition_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.

52.36.6 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.

52.36.7 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:l.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).

52.36.8 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.

52.36.9 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 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 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.

52.36.10 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.

52.36.11 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 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 * * * SERVERROOT/lib/return_job
```

with the same sort of UNIX crontab format argument.

Note: Prior to 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.

52.36.12 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.

52.36.13 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.

52.37 Password and TLS MTA options

There are a few options affecting the MTA's overall handling of authentication, and TLS.

For channel-level configuration, see also TLS and SASL channel options.

And for general Messaging Server configuration of authentication, SSL/TLS use, and certificate handling, see the Auth options, various base.auth*, base.ssl*, and base.tls* options (and for LDAP connections, the ugldapusessl and ldaprequiretls base options), the Base certmap options, and the sectoken options.

52.37.1 plaintextmincipher Option Under mta

If the plaintextmincipher 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.

52.37.2 smtpproxypassword Option

The submit proxy and this option have been removed from the Messaging Server 8.1 (tezpur) release.

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 SmtpProxyPassword in the SmtpProxyAService.cfg file.

52.37.3 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.

52.38 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.

52.38.1 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).

52.38.2 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.

52.38.2.1 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.

52.38.2.2 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.

52.38.2.3 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.

52.38.2.4 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.

52.39 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.

52.39.1 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 imsimta 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_sendopr 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.

52.39.2 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.

52.39.3 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.

52.39.4 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.

52.39.5 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, 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.

52.39.6 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.

52.39.7 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.

52.39.8 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 offical host name of the local channel is used, channel:l.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.)

52.39.9 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.

52.40 Redis MTA options

New in the 8.0.2.3 release, the MTA supports use of the Redis protocol for certain database/storage uses. redis.* options control the MTA's connections to Redis. The *_database_url

MTA options, when set to redis: URL values, configure use of Redis to store MTA databases. See also the enable_sieve_redis MTA option, which enables Sieve filter use of a redis operator in Sieve scripts.

New in the 8.1.0.1 release, support has been added for Redis Sentinel. The redis.servicename option must be specified in addition to configuring Redise Sentinel to enable Sentinel support; when this is done the redis.hostlist and port options should not be specified.

52.40.1 hostlist Option Under redis_client

The hostlist redis option specifies a space separated list of Redis server hosts. The host format is *host*[:*port*]. If *port* is not specified, the port number is determined based on the setting of the redis.port option.

52.40.2 port Option Under redis_client

The portredis option specifies the Redis server default port. If the *port* is not specified in redis.hostlist, the port defaults to this value.

52.40.3 authpassword Option Under redis_client

The authpassword Redis option sets the password that will be used for authentication when communicating with Redis servers.

52.40.4 hostlist Option Under sentinel_client

The hostlist sentinel option specifies a space separated list of Redis Sentinel server hosts. The host format is *host*[:*port*]. If *port* is not specified, the port number is determined based on the setting of the sentinel.port option.

52.40.5 port Option Under sentinel_client

The portsentinel option specifies Redis Sentinel server default port. If the *port* is not specified in sentinel.hostlist, the port number defaults to this value.

52.40.6 authpassword Option Under sentinel_client

The authpassword Redis Sentinel option sets the password that will be used for authentication when communicating with Redis Sentinel servers.

52.40.7 hostlist Option Under redis

The hostlist redis option specifies a space separated list of Redis server hosts. The host format is *host*[:*port*]. If *port* is not specified, the port number is determined based on the setting of the redis.port option.

52.40.8 port Option Under redis

The portredis option specifies the Redis server default port. If the *port* is not specified in redis.hostlist, the port defaults to this value.

52.40.9 authpassword Option Under redis

The authpassword Redis option sets the password that will be used for authentication when communicating with Redis servers.

52.40.10 hostlist Option Under sentinel

The hostlist sentinel option specifies a space separated list of Redis Sentinel server hosts. The host format is *host*[:*port*]. If *port* is not specified, the port number is determined based on the setting of the sentinel.port option.

52.40.11 port Option Under sentinel

The portsentinel option specifies Redis Sentinel server default port. If the *port* is not specified in sentinel.hostlist, the port number defaults to this value.

52.40.12 authpassword Option Under sentinel

The authpassword Redis Sentinel option sets the password that will be used for authentication when communicating with Redis Sentinel servers.

52.41 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, whether using Sieve rules for normally scheduled message expiration (see the expiresieve Message Store option), or subsequent-to-delivery spam/virus filtering using imexpire, see also the External filtering context MTA options.

52.41.1 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.

Since the MTA system filter typically consists of more than one line of text, its value cannot be easily set using the msconfig set command. The `edit filter` command can be used for manual editing of the system filter and the recipe language `set_option` function should be used for automating filter updates.

52.41.2 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

52.41.2.1 Sieve filter interpretation MTA options: `decode_encoded_words` (integer)

RESTRICTED.

Control some decoding of encoded words during Sieve processing. The default value is `7`.

52.41.2.2 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.

52.41.2.3 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.

52.41.2.4 Sieve filter interpretation MTA options: filter_jettison (1 or 2)

New in 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.

52.41.2.5 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.

52.41.2.6 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.

52.41.2.7 Sieve filter interpretation MTA options: sieve_redirect_add_resent (0 or 1)

New in 6.3P1. 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.

52.41.2.8 Sieve filter MTA options: `reject_disables_capture` (0-2)

As of Messaging Server 8.0.1.2, if `reject_disables_capture` is set to 0 (the default), then capture and journal actions will be honored even when a message has been rejected at the sieve level by all recipients. Note that this will prevent promotion of the reject action(s) to an SMTP-level error, causing DSNs to be sent.

Setting this option to 1 will cause capture and journal actions to be canceled in this case, allowing promotion of the error to an SMTP-level response.

As of 8.0.1.3, setting this option to 2 will cause cancellation of any capture and journal actions associated with a rejected recipient address. Each recipient address is considered separately in determining whether or not to cancel these actions. Note that this only applies to capture and journal actions triggered by LDAP attributes; sieve capture and journal actions are not cancelled.

52.41.2.9 Sieve filter MTA options: `discard_disables_capture` (0-2)

As of Messaging Server 8.0.1.3, if `discard_disables_capture` is set to 0 (the default), then capture and journal actions will be honored for recipients for whom the message has been discaded and thus never actually received the mail.

Setting this option to 1 will cause capture and journal associated with such recipients to be canceled.

52.41.2.10 Sieve filter MTA options: `sieve_user_carryover` (0 or 1)

New in 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.

52.41.2.11 Sieve filter MTA options: `sieve_mime_needed` (0 or 1)

MIME message analysis is only performed when necessary and only retains necessary information. However, the serial nature of MTA operations means that subsequent operations may require additional information, forcing the MTA to analyze the message multiple times.

The `sieve_mime_needed` MTA option controls whether or not MIME structure and header information is retained on the first message analysis pass, even if the information is not needed

immediately. A setting of 1 causes the information to be retained, 0 causes it to be discarded. The default is 1.

Saving this information may avoid having to perform another analysis pass. On the other hand, saving the information consumes memory and is wasteful when the information is not likely to be needed.

52.41.2.12 Sieve filter MTA options: `sieve_body_needed` (0 or 1)

MIME message analysis is only performed when necessary and only retains necessary information. However, the serial nature of MTA operations means that subsequent operations may require additional information, forcing the MTA to analyze the message multiple times.

The `sieve_body_needed` MTA option controls whether or not MIME body content information is retained on the first message analysis pass, even if the information is not needed immediately. Note that this information is only used by Sieve body tests. A setting of 1 causes the information to be retained, 0 causes it to be discarded. The default is 0.

Saving this information may avoid having to perform another analysis pass. On the other hand, saving the information consumes considerable memory and is wasteful when no sieve body tests are being performed.

52.41.3 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.

52.41.3.1 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.

52.41.3.2 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.

52.41.3.3 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.

52.41.3.4 Sieve filter limit MTA options: `max_notifys` (integer >= 0)

The `max_notifys` 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_notifys` is checked when processing Sieve "`require`" and "`ihave`" clauses; if such a clause is being applied on "`notify`", the clause will fail if `max_notifys=0` is set.

52.41.3.5 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.

52.41.3.6 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.

52.41.3.7 Sieve filter limit MTA options: `max_sieve_list_size` (0 < integer <= 16,384)

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 16384; (in MS 7.0.5 and earlier, the upper limit was 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`".

52.41.3.8 Sieve filter limit MTA options: `max_sieve_match_iterations` (0 < integer <= 2147483647)

The `max_sieve_match_iterations` MTA option specifies the maximum number of iterations the internal code handling the :matches construct will attempt.

The default is 1,000,000,000; allowed values are integers greater than 0 and less than or equal to 2147483647.

If a :matches operation attempts to perform more than the allowed number of iterations, 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: Too many iterations in :matches`".

52.41.3.9 Sieve filter limit MTA options: `max_sieve_string_size` (0 < integer <= 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.

52.41.3.10 Sieve filter limit MTA options: `max_vacations` (integer >= 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.

52.41.3.11 Sieve filter limit MTA options: `max_variables` (integer >= 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*".

52.41.4 Sieve filter caching MTA options

At the intersection between Sieve filter processing and MTA caching of fetched data, are a couple of MTA options.

52.41.4.1 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.

52.41.5 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.

52.41.5.1 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.

52.41.5.2 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.

52.41.5.3 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".

52.41.5.4 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".

52.41.5.5 Sieve language extension MTA options: enable_sieve_redis (0-2)

New in the 8.0.2.3 release. The enable_sieve_redis MTA option controls whether Sieve filters may use the redis operator to access and manipulate stored data using the redis protocol. A value of 0 disables the use of redis in all Sieve scripts. The default value is 1, meaning that redis may be used in any Sieve script. A value of 2 allows redis to be used in system-level Sieves only.

When enable_sieve_redis is set to 0, then any attempt to use redis will result in a Sieve error "Redis access has been disabled".

When enable_sieve_redis is set to 2, then user-level attempts to use redis will result in a Sieve error "Redis only allowed in system-level sieves".

52.41.5.6 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".

52.41.5.7 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.

52.41.6 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.

52.41.6.1 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.

52.41.6.2 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).

52.41.6.3 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).

52.41.6.4 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.

52.41.6.5 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.

52.41.7 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.

52.41.8 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.

52.41.8.1 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.

52.41.8.2 Transaction logging MTA options: `log_filter` (0-7)

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 or JSON format (`log_format` set to 4 or 5, respectively), Sieve filter action(s) logging, if enabled, appears as the `fl` attribute. The filter action(s) will be enclosed within single quote characters.

Normally the `fl` attribute only appears in XML or JSON format logs if there is an AUTH parameter value to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

As of MS 6.3p1, enabling XML or JSON format (`log_format` set to 4 or 5) 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 enqueuing 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.

52.41.8.3 Transaction logging MTA options: `log_transactionlog` (0-3)

(New in MS 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/JSON attribute name in XML/JSON format logs (`log_format` set to 4 or 5, respectively) is "`tl`". Setting bit 1 (value 2) causes the `transactionlog` string to be included in the `LOG_ACTION` mapping table probe, again at the very end.

Normally the `tl` attribute only appears in XML or JSON format logs if there is transaction log information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

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.*)

52.42 Spamfilter MTA options

The MTA has a number of options affecting Brightmail, ClamAV, Cloudmark, ICAP, Milter (as of 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_optin`*`N`*, `ldap_source_optin`*`N`*, and `ldap_domain_attr_optin`*`N`* MTA options.

For each virus/spam filtering package used, typically at a minimum a pair of options `spamfilter`*`N`*`_library` and `spamfilter`*`N`*`_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 MS 6.2, the MTA supported use of one spam/virus filter package callout (of the site's choice). In MS 6.2, use of up to four spam/virus filter packages callouts was supported (via `spamfilter1_*` through `spamfilter4_*` options). New in 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[1] (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 mentioned spamfilter*_action MTA options and then have the MTA's systemfilter see, consider, and make decisions based upon all the results (all the added header lines).

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, and the error_text_spamfilterN_error MTA options.

[1] 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.

52.42.1 Spamfilter MTA options: optin_user_carryover (bitmask)

New in 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.

52.42.2 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 SERVERROOT/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 (or if making use of Oracle's per-recipient modification Milter extension the new-in-MS-7.0.5.33 `libmilters.so` library image), *e.g.*,

```
msconfig> set mta.spamfilter3_library SERVERROOT/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 SERVERROOT/lib/libicap.so
```

Or for instance, when using the file-drop message archiving plug-in, one of these options would be set to point to the path to the `libarch.so` library image, *e.g.*,

```
msconfig> set mta.spamfilter5_library SERVERROOT/lib/libarch.so
```

52.42.3 Spamfilter MTA options: `spamfilter`*N*`_config_file` (filepath)

The `spamfilter`*N*`_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 special symbolic 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:

- Brightmail spamfilterN_config_file
- ClamAV spamfilterN_config_file
- ICAP spamfilterN_config_file
- Milter spamfilterN_config_file
- SpamAssassin spamfilterN_config_file
- Archive spamfilterN_config_file
- Sieve spamfilterN_config_file

52.42.4 Spamfilter MTA options: spamfilter*N*_name (string)

The spamfilterN_name MTA options are new in 7.0.5.

If spamfilter*N*_library has been specified, but spamfilter*N*_name has not been explicitly set, then the string *N* will be used as the name.

Note that error messages referencing a spamfilter will show this name; that is, error messages that by default include text such as "[slot *N* name *N*]" with a name set will instead include text of the form "[slot *N* name *spamfilterN_name*]".

52.42.5 Spamfilter MTA options: spamfilter*N*_null_optin (string)

Normally, the simple presence of a spam filter opt-in attribute (the attribute named by the ldap_optin*N* MTA option or the attribute named by the ldap_source_optin*N* MTA option in a user entry, or the attribute named by the ldap_domain_attr_optin*N* 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 spamfilter*N*_null_optin MTA options allow for better interaction with such directory tools. A spamfilter*N*_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 opt-in attribute is ignored.

52.42.6 Spamfilter MTA options: spamfilter*N*_action_*M* (URL) spamfilter*N*_verdict_*M* (string)

Each pair of options spamfilter*N*_verdict_*M* and spamfilter*N*_action_*M* for particular values of *N* and *M* specifies 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 MS 6.2 and range between 1 and 8 as of MS 6.3; that is, up to four (as of MS 6.2) or eight (as of 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 spamfilter*N*_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 spamfilter*N*_library and spamfilter*N*_config_file; that is, *N* specifies the (arbitrary) "number" identifying the particular virus/spam filter package. New in 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 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 spamfilter*N*_action_*M* MTA option should also be set, specifying what action to take when the corresponding verdict is returned. The value of a spamfilter*N*_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 52.33 spamfilter*N*_action_*M* MTA option values

Substitution sequence	Description
$U	The verdict name string
$A	The address that this verdict is associated with
$M	(New in MS 6.0) The detailed verdict string (if the spam/virus filter package provided one - not all do)

In 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 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:,$M`" for a `spamfilter`*`N`*`_action_`*`M`* MTA option has some special, short-circuited handling, in that it causes the verdict to be used literally as a Sieve scriptlet itself, omitting the usual URL expansion processing. For "short" verdicts, this is a difference that makes no difference, but it avoids the length limitations imposed during URL expansion that could potentially become relevant for longer verdicts---such as those that milter likes to return. Note that other uses of `$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 `spamfilter`*`N`*`_verdict`/`spamfilter`*`N`*`_action` processing. Thus to achieve some other result for Brightmail clear messages, that is, to get `spamfilter`*`N`*`_verdict`/`spamfilter`*`N`*`_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" `spamfilter`*`N`*`_verdict`/`spamfilter`*`N`*`_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 `spamfilter`*`N`*`_null_action` options controlling behavior when a so-called "null" verdict is returned, and the `spamfilter`*`N`*`_string_action` options controlling behavior when an unrecognized verdict (a verdict without a matching `spamfilter`*`N`*`verdict_`*`M`* value) is returned.

52.42.7 Spamfilter MTA options: `spamfilter`*`N`*`_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 spamfilter*N*_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 spamfilter*N*_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 spamfilter*N*_final MTA options are only available in MS 6.0 and later; iMS 5.2 behaves as if the option had the default value of 0.

In MS 6.2 and later, bit 1 (value 2) of spamfilter*N*_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 spamfilter*N*_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.

52.42.8 Spamfilter MTA options: spamfilter*N*_includeheaders (0 or 1)

(New in 7.0.5) Each spamfilter*N*_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.

52.42.9 Spamfilter MTA options: spamfilter*N*_null_action (URL)

The spamfilter*N*_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.

52.42.10 Spamfilter MTA options: spamfilter*N*_optional (-2, -1, 0, 1, 2, 3, 4)

The spamfilter*N*_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 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. (See the sndopr_priority MTA option for control of the facility and priority of such syslog notices.)

New in 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 MS 6.3) 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.

52.42.11 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.)

52.42.12 Spamfilter MTA options: spamfilter*N*_returnpath (0 or 1)

The spamfilter*N*_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.

52.42.13 Spamfilter MTA options: spamfilter*N*_string_action (URL)

The spamfilter*N*_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 spamfilter*N*_verdict_*M* match for any *M*.

The default for the spamfilter*N*_string_action MTA options is:

```
data:, require "fileinto"; fileinto "$U";
```

Prior to 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 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 MS 6.1.

When using a Milter, the spamfilter*N*_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 `spamfilterN_action_M` MTA options) which means to use the detailed verdict string provided by the milter.

52.43 SPF MTA options

A number of MTA options exist affecting SPF lookups. Note that SPF lookups are enabled via channel options such as `spfhelo`; 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 MS 6.3-0.15.

See also the `error_text_spf_*` MTA options which configure the error text issued in cases of SPF errors.

52.43.1 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.)

52.43.2 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.)

52.43.3 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 spfhelo, spfmailfrom, and spfrcptto channel options. So with spfhelo 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)
```

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
```

52.43.4 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.)

52.43.5 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.)

52.43.6 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 spfhelo, spfmailfrom, and spfrcptto channel options. So with spfhelo 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)
```

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
```

52.43.7 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.)

52.43.8 SPF MTA options: spf_max_recursion (integer)

The spf_max_recursion MTA option's default is 10.

52.43.9 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.)

52.44 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 MS 6.3P1. 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.

The basic steps to set up SRS are as follows:

1. 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 SRS operations for the domain. SRS processing is handled as an overlay on top of normal address processing so nothing pervents a site from using their primary domain as the SRS domain.

2. The `srs_secrets` MTA option must be set to contain at least one SRS secret.

3. The `srs_maxage` can optionally be set to the number of days before an generated SRS address times out and becomes unusable. The default if the option isn't specified is 14 days.

4. Configured SRS usage on the appropriate mail flows. (See below.)

Note that every system that handles email for the selected SRS domain must be configured for SRS processing and must have all three SRS options set identically.

Setting the three options described above is sufficient to enable SRS address decoding. Encoding is another matter - it should only be done to envelope from addresses yoy know are associated with forwarding activity. SRS encoding is controlled by six channel keywords: `addresssrs`, `noaddresssrs`, `destinationsrs`, `nodestinationsrs`, `sourcesrs`, and `nosourcesrs`.

Three conditions have to be met for SRS encoding to occur:

1. The current source channel has to be marked with `sourcesrs`. (`nosourcesrs` is the default).

2. The current destination channel has to be marked with `destinationsrs` (`nodestinationsrs` is the default).

3. The current address, when rewritten, has to match a channel marked `addresssrs` (`noaddresssrs` is the default).

Encoding only occurs when all of these conditions are true. About the simplest setup is a pure forwarding one where all messages enter and exit on the `tcp_local` channel and all nonlocal addresses need SRS handling. In such a setup the `tcp_local` would be marked with the three keywords `sourcesrs`, `destinationsrs`, and `addresssrs`.

See also the `error_text_srs_*` MTA options, which control the exact error text issued when SRS errors occur.

52.44.1 Sender Rewriting Scheme (SRS) controls (`srs_domain`, `srs_hash_algorithmsrs_maxage`, `srs_secrets`)

52.44.1.1 `srs_domain` (domain-name)

(New in MS 6.3P1.) 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.

52.44.1.2 `srs_maxage` (integer)

(New in MS 6.3P1.) 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.

52.44.1.3 `srs_secrets` (comma-separated list of strings)

(New in MS 6.3P1.) 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.

52.44.1.4 `srs_hash_algorithm` (hash-algorithm)

New in MS 8.1.0.3. The `srs_hash_algorithm` MTA option controls what hash algorithm the MTA uses to generate the hash included in SRS addresses. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, SHA256, SHA512, MD128, or MD160. SHA1 is the default. Note that the setting of this option must be the same across a deployment for successful coordination of SRS addresses.

52.44.2 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, `=`.

52.45 Syslog MTA options

The MTA has a number of options relating to generating syslog notices when certain events occur---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.

52.45.1 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 imsimta 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_sendopr 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.

52.45.2 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 MS 6.2 and earlier, the length of the MTA output line sent to syslog is limited to 256 characters. For MS 6.3 and later, the limit is 4096 characters.

The `log_messages_syslog` MTA option operates analogously for MTA message transaction log entries.

52.45.3 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:

```
value = facility * 8 + priority
```

The syslog priority levels and their normal meanings are:

Table 52.34 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 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 52.35 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 (The Syslog Protocol) 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 MS 6.2 and earlier, the length of the MTA output line sent to syslog was limited to 256 characters. For MS 6.3 and later, the limit is 4096 characters.

Note that in MS 6.2, but not in MS 6.3 and later, it was necessary to have the configutil parameter logfile.imta.syslogfacility (corresponding to the Unified Configuration mta.logfile.syslogfacility option) set (*e.g.*, to mail) to have the log_messages_syslog MTA option take effect.

52.45.4 Syslog MTA options: log_sndopr (bitmask)

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. New in MS 8.0, the option takes a bit-encoded value, with bit 0 (value 1) and bit 1 (value 2) having meaning: bit 0 enables syslog notices regarding trouble writing transaction log entries; bit 1 enables syslog notices regarding trouble creating or updating channel counters.

The sndopr_priority option controls the syslog level (facility and severity) of the syslog messages generated.

52.45.5 Syslog MTA options: log_syslog_prefix (bitmask)

New in MS 8.0.2.3. The log_syslog_prefix controls what prefix, if any, is attached to syslog messages generated by the log_connections_syslog and log_messages_syslog MTA options. The default is not to apply a prefix. Note that this behavior differs from previous releases, where the prefix was hard-coded to be "IMTA-W-".

52.45.6 Syslog MTA options: sndopr_prefix (string)

New in MS 8.0.2.3. The sndopr_prefix MTA option sets the prefix attached to MTA messages send to syslog. The default prefix is "IMTA-W-".

52.45.7 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 spamfilter*N*_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). (If the facility has not been explicitly specified in a sndopr_priority setting, as is the case with a value of 5, then the system's default facility is used: usually but not always LOG_USER.)

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.

52.45.8 Spamfilter MTA options: spamfilter*N*_optional (-2, -1, 0, 1, 2, 3, 4)

The spamfilter*N*_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 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. (See the `sndopr_priority` MTA option for control of the facility and priority of such syslog notices.)

New in 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 MS 6.3) 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.

52.46 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:

- Various `logfile` options may be set at the MTA level (`mta.logfile.`*`option-name`* in Unified Configuration) to control logging of insertions into the Message Store performed by `ims-ms` channels and LMTP servers;

- TCP/IP-channel-specific options: among these options are several specifically relating to MTA transaction logging including `MAX_B_ENTRIES`, `MAX_J_ENTRIES`, `MAX_H_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 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 to be written to ignore non-understood attributes, this is not a concern. Therefore, as of MS 6.3P1, 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.

52.46.1 Transaction logging MTA options: log_alternate_recipient (0-3)

(New in MS 8.0.1.) The log_alternate_recipient 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 ABY and ARCPT values to be logged immediately after the SMTP DELIVERBY value is logged, before the intermediate address. "ab" and "al" attributes are used in the XML log format. If bit 1 (value 2) is set in the log_alternate_recipient MTA option, then this information appears as a pair of values in the LOG_ACTION mapping table probe immediately after the DELIVERBY, before the intermediate address.

52.46.2 Transaction logging MTA options: log_auth (0-7)

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, whjich 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 XML or JSON format (log_format set to 4 or 5, respectively).

Normally the au attribute only appears in XML or JSON format logs if there is an AUTH parameter value to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

Enabling XML or JSON format transaction logging, log_format set to 4 or 5, 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.

52.46.3 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)
- Time used by the Indexed Search Converter (ISC) when operating as part of the LMTP server. (RD)
- Time spent creating an LMTP transaction for the store to process; (note that this necessarily includes MTA processing time). (STT)
- Time spent writing the message file(s) to the store. (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,RD
```

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. Finally, zero elements of the comma-separated sublists may be truncated from the right.

In the specific case of spam filter wait timers, an outright spam filter failure is indicated by presence of an "F" suffix followed by an integer code which indicates the phase where the failure occurred. The code values are:

Table 52.36 Spamfilter phase failure codes

Code	Failure	Spamfilter routine	Milter command
1	SYSTEM_FAILURE	bmiInitSystem	n/a
2	MESSAGE_FAILURE	bmiInitMessage	(milter connect)
3	CONNECT_FAILURE	bmiProcessConnection	SMFIC_CONNECT/ SMFIC_HELO
4	MAILFROM_FAILURE	bmiProcessFROM	SMFIC_MAIL
5	AUTHADDR_FAILURE	bmiProcessAUTH	n/a
6	RCPTTO_FAILURE	bmiAccumulateTO/ bmiRejectTO	SMFIC_RCPT
7	ENDRCPTTO_FAILURE	bmiEndTO	SMFIC_DATA
8	HEADER_FAILURE	bmiAccumulateHeaders	SMFIC_HEADER
9	ENDHEADER_FAILURE	bmiEndHeaders	SMFIC_EOH
10	BODY_FAILURE	bmiAccumulateBody	SMFIC_BODY
11	ENDBODY_FAILURE	bmiEndBody	SMFIC_BODYEOB command

12	SUBMIT_FAILURE	bmiEndMessage	SMFIC_BODYEOB response
13	UNKNOWN_FAILURE	n/a	n/a

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"
cd=":::1222,11"
```

52.46.4 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 52.37 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 (log_format is 4), this information appears in the ss attribute within en 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 $T flag (for causing logging) in PORT_ACCESS rejection entries. In XML-compatible format (log_format set to 4), this information is logged in co elements.
2	4	When set, I 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 (log_format is 4), this is the tr attribute.
4	16	When set, C entries may include site-supplied text from a PORT_ACCESS mapping table entry; the text is added to the application information field. (Prior to Messaging Server 7.0, when SMTP server processes began unconditionally probing PORT_ACCESS, this bit had a meaningful side-effect of causing SMTP server processes to always query PORT_ACCESS regardless of channel *sasl* option setting; prior to Messaging Server 7.0, SMTP server processes only queried PORT_ACCESS when the relevant channel was marked maysasl, maysaslserver, mustsasl, or mustsaslserver -- or when this bit was set.)

5	32	When set, include transport information: TCP|*MTA-IP*|*MTA-port*|*remote-IP*|*remote-port* 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.) 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 enqueuing channel when the enqueuing 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.

52.46.5 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 or JSON format tag for this field is "tg". The default is 0 (even in XML or JSON format logging operation).

Normally the tg attribute only appears in XML or JSON format logs if there is conversion tag information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

As of MS 8.0.2.3, bit 3 (value 8) will cause the first conversion tag associated with each message recipient, if present, to be treated as an additional "virtual channel" by the MTA counter

subsystem. This "channel" will then appear in counter output along with all the other channels. Note that no attempt is made to distinguish these virtual channels from normal channels; use of unique names must be dealt with by appropriate configuration.

As of MS 8.1.0.3, bit 4 (value 16) will cause the conversion tags associated with the current message being dequeued to be logged in the extra field of any connection log records.

52.46.6 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.

52.46.7 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 or JSON format (log_format set to 4 or 5, respectively), diagnostic information, if enabled, appears as the di attribute.

Normally the di attribute only appears in XML or JSON format logs if there is diagnostic information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

This option defaults to 1 in order to maintain compatibility with previous releases, where diagnostics information was always logged.

52.46.8 Transaction logging MTA options: log_dkim (0-7)

(New in MS 8.1.0.6.) The log_dkim MTA option controls whether or not the results of any DKIM signing operations are included in transaction log entries and mapping probles. A value of 0 means no DKIM information is extracted. Setting bit 0 (value 1) causes DKIM results to be logged just prior to any smartsend string. The results value consists of a comma-separated list

of entries, one for each active DKIM signing slot. Each entry in turn consists of the slot number and a status string separated by a colon. Status strings beginning with a question mark are error messages, other strings consist of the DKIM identity and selector that were used.

The default for this option is 0 in all log formats except flat JSON (log_format value 6), where it is 1.

The XML/JSON attribute name in XML/JSON format logs (log_format set to 4 or 5-6, respectively) is "fm". Setting bit 1 (value 2) causes from address to be included in the LOG_ACTION mapping table probe, again just prior to the smartsend string.

Normally the dk attribute only appears in XML or JSON format logs if there is DKIM activity to log. However, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

52.46.9 Transaction logging MTA options: log_envelope_id (0-7)

(New in 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 or JSON format (log_format set to 4 or 5, respectively), 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.

Normally the ei attribute only appears in XML or JSON format logs if there is envelope id information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

52.46.10 Transaction logging MTA options: log_filename (0-7)

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 or JSON format (log_format set to 4 or 5, respectively), file name logging, if enabled, appears as the fi attribute. As of MS 6.3p1, enabling XML or JSON format (log_format set to 4 or 5) 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.

Normally the fi attribute only appears in XML or JSON format logs if there is filename information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

52.46.11 Transaction logging MTA options: log_filter (0-7)

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 or JSON format (log_format set to 4 or 5, respectively), Sieve filter action(s) logging, if enabled, appears as the fl attribute. The filter action(s) will be enclosed within single quote characters.

Normally the fl attribute only appears in XML or JSON format logs if there is an AUTH parameter value to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

As of MS 6.3p1, enabling XML or JSON format (log_format set to 4 or 5) 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 enqueuing 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.

52.46.12 Transaction logging MTA options: log_format (1-6)

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 MS 6.3) requests XML-compatible format.

A value of 5 (new in MS 8.0.2.3) requests JSON-compatiable format. And finally, a value of 6 (new in MS 8.1.0.2) specifies "flat" JSON format.

As of MS 6.3p1, note that setting log_format to a value greater than 3 (XML or JSON format) 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 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).

In the (new in MS 8.0.2.3) JSON-compatible format (log_format set to 5), each message log entry appears as a single JSON object containing multiple name-value pairs. All values are strings or integers; at present there are no arrays or nested objects. The first name-value pair always has a name of "ty" and its value specifies the object type. Three object types 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).

The (new in MS 8.1.0.2) value of 6 also creates a JSON-compatible format, but one which attempts to avoid both JSON structure and structured field content, as well as making the format compatible for consumption by Lumberjack. Specifically, the following changes are made:

- The ts timestamp attribute is encoded as integer number of milliseconds since the epoch.
- The tl transactionlog attribute is renamed to msg and is included in every log entry.
- The tg conversion tag attribute is broken up into as many as 10 separate tag attributes t0, t1, t2, ..., each containing a separate tag. The eleventh and subsequent tags, if they exist, are not logged.
- The he array of header field values is broken up into as many as 10 separate header attributes h0, h1, h2, ..., each containing a separate header field value. The eleventh and subsequent header fields, if they exist, are not logged.
- The so attribute contains the envelope from address. The domain from this address is broken out and appears in an additional sd attribute.
- The various protocol modifiers to the ac action attribute are moved to a separate sp attribute.
- When the tr transport information attribute contains TCP address information it is replaced by li and ri attributes containing the local and remote IP addresses, respectively. Port information is not logged.
- The ap application information attribute is replaced by a po attribute containing the protocol and, if SSL/TLS is being used, a cs attribute containing the TLS/SSL information string. Other parts of the application information string are not logged.

Note that the maximum length of MTA transaction record, both for message transaction records and connection transaction records and regardless of format, is 4096 characters.

52.46.12.1 Transaction entries

Message transaction (en) elements/objects can have the following attributes (in XML) or name-value pairs (in JSON):

- ts - time stamp (always present) Note: The format changes from ISO 8601 format to a UNIX epoch value in milliseconds if log_format is set to 6
- 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)
- sp - protocol modifier values from action field (Separated out from the action field if log_format is set to 6)
- 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)
- sd - domain from source address (present if log_format is set to 6)
- 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 and optionally the message IDs of related messages, separated by commas (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, transport information is available, and log_format is not set to 6)
- li - local IP address (present if bit 5/value 32 of log_connection is set, transport information is available, and log_format is set to 6)
- ri - remote IP address (present if bit 5/value 32 of log_connection is set, transport information is available, and log_format is set to 6)
- ap - application information (present if bit 6/value 64 of log_connection is set, application information is available, and log_format is not set to 6)
- po - application protocol (present if bit 6/value 64 of log_connection is set, application information is available, and log_format is set to 6)

- po - SSL/TLS information string (present if bit 6/value 64 of log_connection is set, application information is available, SSL/TLS is being used, and log_format is set to 6)
- 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) Note: This changes from a single tg attribute to per-tag t0, t1, ... attributes if log_format is set to 6
- 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)
- sm - smartsend information(present if bit 0/value 1 of log_smartsend is set and smartsend information is available)
- dk - (new in 8.1.0.6) DKIM signing information (present if bit 0/value 1 of log_dkim is set and DKIM signing was enabled)
- fm - (new in 8.1.0.2) Address from header From: field (present if bit 0/value 1 of log_from is set and a parsable From: field address is present)
- tl - (new in 8.0) String produced by "transactionlog" Sieve actions Note: The attribute name changes from tl to msg if log_format is set to 6
- he - (new in 8.1.0.1) Selected header fields (present if bit 2/value 4 of log_header is set and any of the selected headers are present) Note: This changes from a single he attribute to per-field h0, h1, ... attributes if log_format is set to 6

Here are two sample en entries in XML format showing various different fields being logged (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="l" ac="E" sz="12" so="info-E8944AE8D033CB92C2241E@whittlesong.com"
od="rfc822;ned+2Bcharsets@mauve.sun.com"
de="ned+charsets@mauve.sun.com" rf="22"
fi="/path/ZZ01LI4XPX0DTM00IKA8.00" ei="01LI4XPQR2EU00IKA8@mauve.sun.com"
mi="&lt;11a3b401c4dd01$7c1c1ee0$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"/>

<en ts="2016-08-30T06:28:59.93" pi="28e3d.4.233" sc="tcp_local"
dc="process" ac="E" sz="5" so="" od="rfc822;user+2Berrors@example.com"
de="user+errors@example.com" rf="276" ei="01QWOCSBWTPO003L8D@example.com"
mi="&lt;01QWOCSEZD@example.com&gt;,&lt;1658a875@example.net&gt;"
ss="TCP-DAEMON.example.com" se="-1" tr="" ap="" qt="0" df="0"
cd=",,,47,,20:83;24,,,,,,::406,17"/>
```

And here is a sample field in JSON format (also wrapped for clarity):

```
{"ty":"en","ts":"2018-10-16T07:14:35.35","pi":"547a.3.3","sc":"tcp_intranet",
"dc":"tcp_local","ac":"E","sz":1,"so":"sender@example.com",
"od":"rfc822;recip@example.net","de":"recip@example.net","rf":20,
"fi":"/opt/sun/comms/messaging64/data/queue/tcp_local/003/ZZk0W5l0MfgU0.00",
"mi":"<0PGP00G053JVOQ00@multke.example.org>","us":"mailsrv",
"ss":"[127.0.0.1] ([127.0.0.1])","pr":3,"in":"recip@example.net","qt":0,
"df":68,"cd":":0,,,0,,,::1567,0"}
```

52.46.12.2 Connection entries

Connection transaction (`co`) entries can have the following attributes/name-value pairs:

- `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) (new in MS 8.1.0.3) in connection open/close records, conversion tags associated with the current message being dequeued (present only if bit 4 (value 16) is set and there are conversion tags to log)
- `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/connection was open (present only if bit 0/value 1 of `log_queue_time` is set; specific to connection entries)

Here is a sample `co` entry in XML format (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="O" tr="TCP|209.55.107.55|25|209.55.107.104|33469" ap="SMTP"/>
```

And here is a similar entry in JSON format: (also wrapped for clarity):

```
{"ty":"co","ts":"2018-10-16T07:14:09.27","pi":"547a.3.0","sc":"tcp_local",
"dr":"+","ac":"O","tr":"TCP|127.0.0.1|25|127.0.0.1|48023","ap":"SMTP"}
```

52.46.12.3 Header entries

Header (he) entries have the following attributes/name-value pairs:

- 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 in XML:

```
<he ts="2004-12-08T00:38:31.41" pi="1074b3.61.281" va="Subject: foo"/>
```

And one in JSON:

```
{"ty":"he","ts":"2018-10-16T07:14:35.35","pi":"547a.3.3",
 "va":"Subject: This is a test"}
```

52.46.13 Transaction logging MTA options: log_from (0-7)

(New in MS 8.1.0.2.) The log_from MTA option controls whether or not the address in the header From: field is extracted and used in transaction log entries and mapping probles. A value of 0 means the From: field is not extracted. Setting bit 0 (value 1) causes from address to be logged just prior to any DKIM string. The XML/JSON attribute name in XML/JSON format logs (log_format set to 4 or 5-6, respectively) is "fm". Setting bit 1 (value 2) causes from address to be included in the LOG_ACTION mapping table probe, again just prior to the DKIM string.

The default for this option is 0 in all log formats except flat JSON (log_format value 6), where it is 1.

Normally the fm attribute only appears in XML or JSON format logs if there is a from address to log. However, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

52.46.14 Transaction logging MTA options: log_futurerelease (0-3)

(New in 8.0.) The log_futurerelease MTA option controls whether or not any SMTP 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.

52.46.15 Transaction logging MTA options: log_header (0-7)

The log_header MTA option controls whether the MTA writes message headers to the message transaction log file, mail.log. Or, as of 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 MS 6.1, the option takes a bit-encoded integer value instead, with the lowest two bits defined as a group. A value of 1 for these bits 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. As of MS 8.0.2, a value of 3 for the lowest two bits is also supported, which enables header logging during dequeues but not enqueues.

As of MS 8.1.0.1, bit 2, value 4, is also defined. If set it causes header logging to be included in the transaction log entry itself. A value of 0 logs header information in separate records.

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 in legacy configuration or unified configuration prior to MS 8.1.0.1. As of MS 8.1.0.1, the log_header_options MTA option should be used to control which headers are written. The format of this file/option 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 he elements.

When enabling log_header, consider also enabling log_process, as it is helpful for correlating header entries with corresponding regular message entries.

If the goal is to record subsets of information from one or more header lines, rather than necessarily logging full header lines, see as an alternative to log_header the (both new-in-MS-8.0) log_transactionlog MTA option and Sieve "transactionlog" action.

52.46.16 log_header_options Option

The log_header_options MTA option contains the entire header options file controlling the logging of header fields. It corresponds to the legacy configuration log_header.opt file.

The log_header_options MTA option would typically be set or modified by using the edit command of msconfig, *e.g.*:

```
msconfig> edit log_header_options
```

52.46.17 log_headers_maxchars Option

New in MS 8.1.0.1, the log_headers_maxchars MTA options places a limit on the size of the header lines included in primary transaction log entries as a result of the actions of the AUTH_REWRITE or AUTH_ACCESS mapping. THe default is 200 bytes.

52.46.18 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.

52.46.19 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.

Bits 0 (value 1) and 1 (value 2) of the logged field roughly correspond to those in the deliveryflags channel option. The other bits in this value are intentionally undocumented.

52.46.20 Transaction logging MTA options: log_intermediate (0-63)

(New in 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 52.38 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 or JSON format (log_format set to 4 or 5, respectively), the intermediate address is the in 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 or JSON format (log_format set to 4 or 5, respectively), the initial address is the ia 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.
3	8	New in 8.0.2.3. When set, the in attributes appears in XML and JSON logs unconditionally, even when there is no intermediate address information to log.
3	8	New in 8.0.2.3. When set, the ia attributes appears in XML and JSON logs unconditionally, even when there is no initial address information to log.

52.46.21 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.

52.46.22 Logging Indexed Search Converter status: log_isc_status (bitmask)

New in MS 8.0.2. The log_isc_status MTA option causes a field recording information sent to or reported by the Indexed Search Converter (ISC) to be included in log file entries (both "S" and "J" entries on LMTP server systems, "E" entries on front end systems) 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 "diagnostics" field (log_diagnostics option) and before the transport

information filed (log_transportinfo option). The XML format tag for this field is "is". The default is 1 in XML and JSON format logs and 0 in other formats.

Normally the is attribute only appears in XML or JSON format logs if there is ISC status information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

The content of the "is" field differs depending on the type of system performing the logging. On LMTP server systems the field will begin with an indication of whether or not immediate conversion was requested - "IMMEDIATE" if it was, "DEFERRED" if it wasn't. This will be optionally followed by any status information returned by the converter.

On all other types of systems the "is" will contain when the indexing operaton is supposed to be performed if a preference was selected by the system doing the logging. A value of "DEFERRED" is recorded if deferred indexing was requested. A value of "IMMEDIATE-*r*-*t*, where *r* is the retry limit and *t* is the timeout, is recorded when immediate indexing is requested.

52.46.23 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 type of information, which can be useful when there is a need to correlate a message in the store with MTA actions. Prior to MS 8.0 the field consisted of these two values delimited by a colon. in MS 8.0 and later release two additional fields have been added: One for the message digest and another for the name of the folder the message is actually delivered to. Note that the message digest is only included when one is calculated for other reasons.

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 or JSON log format (log_format set to 4 or 5, respectively). 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.

Normally the mu attribute only appears in XML or JSON format logs if there is mailbox uid information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

It's a simple matter to log this information in the case of ims-ms channel. But in the case of LMTP things are not so simple - the information naturally exists in the LMTP server but not the client. An LMTP extension is needed to pass the information to the client for logging purposes.

Consequently, as of MS 8.0.1.2, the LMTP client and server have been enhanced to support a new UID extension. This extension, if present, provides a single UID parameter on the MAIL FROM command that accepts the values "NO" or "RET". If the latter is specified the final LMTP responses to DATA/BDAT will include the UID, UIDVALIDITY, digest value (if available), and optionally the folder if different from INBOX, separated by colons and enclosed in angle brackets. For example:

```
S: 220 multke.mrochek.com -- Server LMTP (Oracle Communications Messaging Server 8.0.1.2 64bit (built Sep 19 2015))
C: LHLO multke.mrochek.com
S: 250-multke.mrochek.com
S: 250-8BITMIME
S: 250-UID
S: 250-PIPELINING
S: 250-CHUNKING
S: 250-XDFLG
S: 250-XQUOTA
S: 250-XAFLG
S: 250-ENHANCEDSTATUSCODES
S: 250-HELP
S: 250 SIZE 0
C: MAIL FROM:<> UID=RET
S: 250 2.5.0 Address Ok.
C: RCPT TO:<test1+folder@ims-ms-daemon> XDFLGS=5
S: 250 2.1.5 test1@ims-ms-daemon OK.
C: RCPT TO:<test2@ims-ms-daemon>
S: 250 2.1.5 test2@ims-ms-daemon OK.
C: DATA
S: 354 Enter mail, end with a single ".".
C: Subject: Test message
C:
C: This is a test.
C: .
S: 250 2.5.0 <1445028362:2::folder> Delivery to user OK
S: 250 2.5.0 <1440097745:2:> Delivery to user OK
```

This extension is enabled by default and cannot be disabled. The LMTP client will use if it present to obtain UID information to use in conjunction with the `log_mailbox_uid` MTA option as well as the recall facility.

Additionally, as of MS 8.0.1.2 the `log_mailbox_uid` MTA option now enables logging of UID information in the S records logged by the LMTP server. Note that this logging does not depend on use of the LMTP extension.

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.

52.46.24 Transaction logging MTA options: `log_message_id` (0-7)

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 or JSON format (`log_format` set to 4 or 5, respectively), message ID logging, if enabled, appears as the `mi` attribute. As of MS 6.3p1, enabling XML or JSON format (`log_format` set to 4 or 5) 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.

Multiple message IDs can appear, separated by commas. When mutiple IDs are present, the first is the ID of the message being logged and subsequent IDs are the IDs of related messages, e.g., when a DSN or MDN is generated and enqueued by the MTA, the second message ID will be that of the message the DSN or MDN is in regards to.

Normally the `mi` attribute only appears in XML or JSON format logs if there is message id information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

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.

52.46.25 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.

52.46.26 Transaction logging MTA options: log_node (0 or 1; OpenVMS only until 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 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 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.)

52.46.27 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 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 52.39 log_notary MTA option bit value

Bit	Value	Usage
0	1	Use header-style delivery receipt requests, corresponding to the use of the reportheader channel option or reportboth 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.e.*, NOTARY parameter NOTIFY=FAILURE.
3	8	DSN delivery receipt requested: Send a DSN (delivery receipt) if message delivery succeeds, *i.e.*, 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.e.*, NOTARY parameter NOTIFY=DELAY.
5	32	Internal flag keeping track of notification handling for mailing lists.
6	64	Suppress all notifications, *i.e.*, 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.

52.46.28 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 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.

52.46.29 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 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 is 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) sequence number that uniquely identifies the logging operation as a whole. That is, in the case of a single threaded channel

process-id.sequence

or in the case of a multithreaded channel

process-id.thread-id.sequence

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. The sequence number provides the ability to identify all the entries associated with the processing a a specific queue entry.

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.*

52.46.30 Transaction logging MTA options: log_queue_time (0-3)

(New in 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 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, log_remote_mta remote-MTA 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.

As of 6.3P1, setting bit 0/value 1 of log_queue_time also enables logging of connection time information in "O" (time to open conection), "Y" (time spent before open failure), and

"C" (time connection was open) connection log records. If XML format logging is enabled this information appears in the ct 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 and/or connection time information to be included in LOG_ACTION mapping probes for both transaction and connection log records.

52.46.31 Transaction logging MTA options: log_reason (0-3)

(New in 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 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 or JSON format (log_format set to 4 or 5, respectively), the reason logging, if enabled, appears as the re attribute.

Normally the re attribute only appears in XML or JSON format logs if there is reason information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

New in 7.0-3.01, bit 1 (value 2), if set, causes reason information to be included in LOG_ACTION mapping probes.

52.46.32 Transaction logging MTA options: log_remote_mta (0-3)

(New in MS 8.0.2.3.) The log_remote_mta MTA option controls whether the remote MTA name is included in MTA message transaction log entries as a separate field and LOG_ACTION mapping probes that correspond to an operation involving a remote SMTP or LMTP server.

Setting bit 0/value 1 enables the generation of the remote MTA and also disables the inclusion of this information in the diagnostics field; clearing the bit disables the use of a separate field. The default is 0.

If logging is enabled, the remote MTA name will be logged just after the SMTP delivery status (log_diagnostics SMTP diagnostic) field and just before the ISC status field. In XML or JSON format (log_format set to 4 or 5, respectively), the reason logging, if enabled, appears as the rm attribute.

Normally the rm attribute only appears in XML or JSON format logs if there is remote MTA information to log. Setting bit 2 (value 4) will cause the attribute to appear unconditionally.

Bit 1 (value 2), if set, causes remote MTA information to be included in LOG_ACTION mapping probes.

52.46.33 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 52.40 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.

52.46.34 Transaction logging MTA options: log_smartsend (0-7)

(New in MS 8.1.0.1.) The log_smartsend MTA option controls whether information about the actions of smartsend callouts are included in MTA message transaction log records. The option defaults to 0, meaning that this information isnot logged. Setting bit 0 (value 1) causes smartsend information to be logged just prior to any header from string. The XML/JSON attribute name in XML/JSON format logs (log_format set to 4 or 5, respectively) is "sm". Setting bit 1 (value 2) causes smartsend information to be included in the LOG_ACTION mapping table probe, again just prior to the header from string.

Normally the sm attribute only appears in XML or JSON format logs if there is transaction log information to log. However, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

As of MS 8.1.0.6 the DEQUEUE_ACCESS mapping and the AUTH_ACCESS mapping provide the means to append strings to the smartsend log value.

52.46.35 Transaction logging MTA options: log_times (0-63)

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 or JSON format (log_format set to 4 or 5, respectively), 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.

Normally the dd attribute only appears in XML or JSON format logs if there is deferred delivery time information to log. As of MS 8.0.2.3, setting bit 4 (value 16) will cause the attribute to appear unconditionally.

Bit 2 (value 4), if set, enables logging any message expiry time. In XML or JSON format (log_format set to 4 or 5, respectively), 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.

Normally the dd attribute only appears in XML or JSON format logs if there is expiry date information to log. As of MS 8.0.2.3, setting bit 5 (value 32) will cause the attribute to appear unconditionally.

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.

52.46.36 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 XML or JSON format (log_format set to 4 or 5, respectively).

Normally the mt attribute only appears in XML or JSON format logs if there is tracking information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

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.

52.46.37 Transaction logging MTA options: log_transactionlog (0-3)

(New in MS 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/JSON attribute name in XML/JSON format logs (log_format set to 4 or 5, respectively)

is "tl". Setting bit 1 (value 2) causes the transactionlog string to be included in the LOG_ACTION mapping table probe, again at the very end.

Normally the tl attribute only appears in XML or JSON format logs if there is transaction log information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

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.*)

52.46.38 Transaction logging MTA options: log_uid (0-7)

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 or JSON log format (log_format set to 4 or 5, respectively). 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.

Normally the ui attribute only appears in XML or JSON format logs if there is uid information to log. As of MS 8.0.2.3, setting bit 2 (value 4) will cause the attribute to appear unconditionally.

52.46.39 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.

52.46.40 Transaction logging MTA options: log_username (0-511)

The log_username MTA option is a bit-encoded value which controls whether or not user information 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.)

The bits in the log_username option are described in the following table:

Table 52.41 log_username option bit values

Bit	Value	Usage
0	1	Enables authenticated username logging in transaction log entries. (Username logging in connection log "U" records is always enabled.) In the MTA message transaction log entries the username will be logged after the final form envelope To address field - 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 appears in the us attribute.
1	2	(New in 7.0-3.01) Causes username information to be included in LOG_ACTION mapping probes.
2	4	(New in MS 8.0) Enables inclusion of the primary mail address associated with the authenticated username in the transaction log. This information appears immediately after the username and under the XML attribute "as".
3	8	(New in MS 8.0) Enables inclusion of the primary mail address associated with the authenticated username in LOG_ACTION mapping table probes. This address appears immediately after the username in the mapping table probe.
4	16	(New in MS 8.0) Causes additional information associated with an authentication operation to be logged (currently just the name of the authentication mechanism), XML tag "ex"; additional information may be added in the future) to be included in connection log "U" records;
5	32	(New in MS 8.0) Causes the same additional information described under the the previous bit to be included in LOG_ACTION mapping table probes immediately after the authenticated username.

6	64	(New in MS 8.0.2.1) Causes the administrator's username, in the case of administrative proxy authentication to be logged in "U" connection log records immediately after the additional authentication information field and using the XML tag "ua".
7	128	(New in MS 8.0.2.1) Causes the administrator's username, in the case of administrative proxy authentication to be included in LOG_ACTION mapping table associated with "U" connection log records. The field appears immediately after the additional authentication information field.
8	256	(New in MS 8.0.2.3) Include the "us" attribute in XML or JSON format transaction entries, regardless of whether or not there's user name information to log.
9	512	(New in MS 8.0.2.3) Include the "as" attribute in XML or JSON format transaction entries, regardless of whether or not there's user name information to log.

Bit 0 is the least significant bit.

As of MS 6.3p1, enabling XML or JSON format (log_format set to 4 or 5, respectively) 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.

52.46.41 Transaction logging MTA options: log_8bit_encode 0 or 1)

The log_8bit_encode MTA option controls how non-ASCII material is written to XML format MTA transaction log files. A value of 0 causes such material to be written as-is. Any nonzero value will cause all valid UTF-8 characters to be written as XML numeric character references.

How non-ASCII octets that does not begin a valid UTF-8 sequence is controlled by specific values of the log_8bit_encode MTA option. Currently defined values are:

1. Write the octet as a XML numeric character reference. Note that this is tantamount to interpreting the octet as a character in the ISO-8859-1 charset.
2. Write the octet as an XML entity reference of the form &raw-xx, where xx is the lower-case hexadecimal representation of the octet.
3. Write out a space in place of the octet.

Note that this setting only applies to XML format logs (log_format = 4); non-ASCII characters are always written as-is when other formats are used.

52.46.42 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.

52.46.43 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 *N*th 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.

52.46.44 Transaction logging MTA options: return_cleanup_period (integer)

Sites may supply a cleanup script, DATAROOT/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 *N*th run of the return job, set the value of the return_cleanup_period MTA option to *N*.

52.47 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.

52.47.1 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.

52.47.2 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.

52.47.3 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.

52.47.4 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".

52.47.5 mail_delivery_filename Option

OpenVMS only.

52.47.6 missing_address Option

OpenVMS only.

52.47.7 multinet_mm_exclusive Option

OpenVMS only.

52.47.8 OpenVMS user agent MTA options: read_receipt_off (string)

OpenVMS only.

The read_receipt_offMTA 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.

52.47.9 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.

52.47.10 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.

52.47.11 use_mail_delivery Option

OpenVMS only.

52.47.12 vms_mail_exclusive Option

OpenVMS only.

Chapter 53 MTA Tailor options

In legacy configuration, various MTA installation and operational parameters would be set in the MTA Tailor file (`imta_tailor`) 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 (either `/var`/SERVERROOT or SERVERROOT/`data`) and CONFIGROOT value (either `/var`/SERVERROOT/`config` or 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.

53.1 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.

53.1.1 MTA Tailor options: `imta_root` (directory path)

DELETED. Root of the MTA installation is now determined via the SERVERROOT environment variable.

Note that for MTA purposes, a special symbolic name IMTA_ROOT is still available for use in certain contexts (such as when the MTA is interpreting msconfig option values, or file path specifications).

53.1.2 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, namely *SERVERROOT*/lib/.

Note that for MTA purposes, a special symbolic name IMTA_LIB is still available for use in certain contexts, such as when specifying the location of an Oracle-supplied shared library being used in a mapping callout or rewrite rule callout.

53.1.3 MTA Tailor options: imta_bin (MTA directory path)

DELETED.

The directory containing MTA executable binaries is no longer configurable, instead now being determined via the SERVERROOT, namely *SERVERROOT*/lib/.

Note that the special symbol name IMTA_BIN is still available for use in some MTA contexts.

53.1.4 MTA Tailor options: imta_table (MTA directory path)

DELETED.

The location of the MTA configuration directory is now determined using the CONFIGROOT. Note that with Unified Configuration, many fewer files comprise the MTA configuration than formerly, with legacy configuration, so this *CONFIGROOT*/ directory contains fewer files.

Note that the special symbol name IMTA_TABLE is still available for use in some MTA contexts.

53.1.5 MTA Tailor options: imta_dl (MTA directory path)

DELETED.

Obsolete directory used by iMS 5.x for directory synchronization. No longer used.

53.1.6 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.

53.1.7 imta_tmp Option

The legacy configuration imta_tmp MTA Tailor option has been replaced in Unified Configuration by tmpdir.

(Note that in MS 7.0, the imta_tmp MTA Tailor option was replaced by the tmpdir option in the MTA option file.)

53.1.8 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 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 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/.

53.1.9 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.

Note that for MTA purposes, a special symbolic name IMTA_QUEUE is still available for use in certain contexts (such as when the MTA is interpreting msconfig option values, or file path specification input to MTA command line utilities).

53.1.10 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.

53.1.11 imta_lang Option

The legacy configuration imta_lang MTA Tailor option has been replaced in Unified Configuration by the langdir MTA option, mta.langdir.

53.1.12 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 *DATAROOT*/site-programs/. If location customization is desired, use symbolic links.

Note that the special symbol name IMTA_PROGRAM is still available for use in some MTA contexts.

53.2 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.

53.2.1 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 *CONFIGROOT*/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.

53.2.2 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 in legacy configuration as *CONFIGROOT*/imta.filter; in Unified Configuration, see instead the systemfilter MTA option.

53.2.3 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 CONFIGROOT/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 *CONFIGROOT*/config.xml.

53.2.4 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 *CONFIGROOT*/config.xml

53.2.5 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 *CONFIGROOT*/advanced/command_data.

53.2.6 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 *CONFIGROOT*/advanced/config_data.

53.2.7 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 *CONFIGROOT*/advanced/charset_data.

53.2.8 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 CONFIGROOT value, namely *CONFIGROOT*/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.

53.2.9 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 *DATAROOT*/log/mail.log_current.

53.2.10 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 *DATAROOT*/log/mail.log_yesterday.

53.2.11 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 *DATAROOT*/log/mail.log.

53.2.12 imta_primary_connection_log Option

DELETED.

Name and location of today's MTA connection transaction log file. No longer configurable, and instead located via the SERVERROOT environment variable as *DATAROOT*/log/connection.log_current.

53.2.13 imta_secondary_connection_log Option

DELETED.

Name and location of yesterday's MTA connection transaction log file. No longer configurable, and instead located via the SERVERROOT environment variable as *DATAROOT*/log/connection.log_yesterday.

53.2.14 imta_tertiary_connection_log Option

Name and location of the MTA's cumulative connection transaction log file. No longer configurable, and instead located via the SERVERROOT environment variable as *DATAROOT*/log/connection.log.

53.2.15 imta_name_content_file Option

Table of conversion channel part reading defaults by media type and file extensions. The name and location of this file is no longer configurable, and instead located via the SERVERROOT environment variable as *SERVERROOT*/lib/name_content.dat.

53.2.16 imta_forward_data Option

The DELETED imta_forward_data MTA Tailor option had been used to specify the name and location of the 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; the text form source file is now unconditionally IMTA_TABLE:forward.txt (where note that IMTA_ROOT is a special symbolic name interpreted by the MTA).

53.2.17 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; the text form source file is now unconditionally IMTA_TABLE:reverse.txt (where note that IMTA_ROOT is a special symbolic name interpreted by the MTA).

53.2.18 imta_general_data Option

DELETED: Location no longer configurable. (The imta_general_data MTA Tailor option had been used to specify the location and name of the text file source for the MTA general database, used when the use_text_database MTA option specifies use of such "text" form, replacing the deprecated crdb form of the general database.) The text form source file is now unconditionally IMTA_TABLE:general.txt (where note that IMTA_ROOT is a special symbolic name interpreted by the MTA).

53.2.19 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".

53.2.20 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.

53.2.21 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.

53.2.22 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.

53.2.23 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.)

53.2.24 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 no longer supported ssr: URLs.)

53.2.25 imta_dn_to_id_database Option

DELETED: the imta_dn_to_id_database MTA Tailor option had specified the name and location of the legacy dirsync DN to ID database.

53.2.26 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.)

53.2.27 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.)

53.2.28 imta_mapping_file Option

DELETED: This location is not longer configurable. (The imta_mapping_file MTA Tailor option had been used to specify the name and location of the file containing MTA mapping tables.)

In legacy configuration, the mappings file is now always *CONFIGROOT*/mappings. In Unified Configuration, the content of the mappings file has been incorporated into the XML configuration file, config.xml, as named mapping groups.

53.2.29 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; in Unified Configuration, see instead the conversions MTA option.)

53.2.30 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.

53.2.31 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.

53.2.32 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.

53.2.33 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.

53.2.34 imta_libmap Option

Formerly, the name and location of the imtamap 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 imtamap shared library is no longer configurable.

53.2.35 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.

53.3 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.

53.3.1 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).

53.3.2 Tailor file options: imta_user_username (string)

DELETED.

User name for untrusted operations such as MTA sequence number file access, or untrusted pipe channel operations. Moved to the restricted.cnf file as the "noprivuser" option.

53.3.3 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.

53.4 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.

53.4.1 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.) In Unified Configuration, see instead the schedule.task:purge.crontab option.

53.4.2 imta_version_limit_period Option

Obsolete option. In Unified Configuration, see instead the schedule.task:purge.crontab option.

53.4.3 imta_return_period Option

Obsolete option. In Unified Configuration, see instead the schedule.task:return_job.crontab option.

53.4.4 imta_return_synch_period Option

Obsolete option. See the synch_time Job Controller option instead.

53.4.5 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.

53.4.6 imta_return_cleanup_period Option

The legacy configuration imta_return_cleanup_period MTA Tailor option has been replaced in Unified Configuration by return_cleanup_period MTA option, mta.return_cleanup_period.

53.4.7 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.

53.4.8 imta_synch_cache_period Option

Obsolete option. See the synch_time Job Controller option instead.

Chapter 54 Dispatcher

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.

54.1 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.

54.1.1 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.)

54.2 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
```

In legacy configuration mode, Dispatcher options are set in the `dispatcher.cnf` file.

Generally, the Dispatcher must be restarted in order for its option changes to take effect.

When the `dispatcher.use_nslog` option has been enabled, 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.

54.2.1 `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.

54.2.2 `enable` Option Under service

(New in 7.0.5) The `enable` Dispatcher `service` option controls whether the service is enabled. It defaults to `1`, but may be set to `0` to disable a service.

54.2.3 `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.

54.2.4 debug Option Under dispatcher

The `debug` Dispatcher option (available under `dispatcher` and `service`) enables debug output from the Dispatcher. It may be set either directly at the `dispatcher` level, or may be enabled or disabled more selectively for a specific Dispatcher service by setting it under a named `service`.

The `IMTA_DISPATCHER_DEBUG` environment variable can also be used to control dispatcher debugging.

Table 54.1 Dispatcher Debug Bit Mask Values

Bit	Value	Description
0-31	-1	Extremely verbose output
0	1	Startup, initialization, shutdown message
1	2	Thread increment/decrement messages
2	4	Configuration loading messages
3	8	Process creation messages
4	16	Process activity messages
5	32	PORT_ACCESS mapping, connection messages
6	64	Process calculation messages
7	128	Process shutdown messages
8	256	Socket listen, cookie messages
9	512	Dispatcher internal I/O messages
10	1024	Dispatcher internal read messages
11	2048	Currently unused
12	4096	Process management messages
13	8192	Process handoff messages
14	16384	Dispatcher message processing messages
15	32768	Currently unused
16	65536	Successful connection message
17	131072	Connection accept, TLS messages
18	262144	Currently unused
19	524288	Commands
20	1048576	Statistics messages
21	2097152	Currently unused
22	4194304	Currently unused
23	8388608	Currently unused
24	16777216	Connection rejection message (subset of bits 4 and 5)

54.2.5 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 service option specifies the host name or IP address of source against which to check incoming connections.

Note that an alternative to use of the dns_verify_domain Dispatcher service option is use of a dns_verify routine callout from a mapping table such as PORT_ACCESS. The dns_verify_domain Dispatcher service option is simple to set -- but use of dns_verify

callouts (which come in several flavors) from a mapping table allows for more precise control of checks.

Note that PORT_ACCESS mapping table probes (which may optionally be configured to perform their own DNS verification checks using a dns_verify routine callout) are made before any dns_verify_domain Dispatcher service option lookups are consulted. If a PORT_ACCESS probe rejects a connection, then the dns_verify_domain Dispatcher service option lookup does not need to be (and is not) performed. And as of 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 Dispatcher debugging and log files. Such dispatcher.log entries will take the form:

```
access_control: host a.b.c.d found on DNS list and rejected
```

54.2.6 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. This option is available both directly at dispatcher level, or may be set under named Dispatcher services to override for that service the general Dispatcher value.

Prior to 7.0.5, the default value had been 120 (2 minutes) but initial configuration set this value to 0 (no history retained). As of 7.0.5, the default value is 0.

54.2.7 image Option

The image Dispatcher service 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.

54.2.8 interface_address Option

The legacy option interface_address (Job Controller option and Dispatcher option) is an alias for the job_controller.listenaddr and dispatcher.listenaddr options in Unified Configuration.

54.2.9 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.

54.2.10 listenaddr Option Under service

The listenaddr Dispatcher service 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.

54.2.11 logfilename Option

Specifying the logfilename option for a named 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 value may also make use of special symbolic names known to the MTA, such as IMTA_LOG.) 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".

54.2.12 max_conns Option Under dispatcher

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.

54.2.13 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 is 50. In Messaging Server 7.0.2 and earlier as well as legacy 32 bit platforms, 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.

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.

54.2.14 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).

54.2.15 max_handoffs Option

The max_handoffs Dispatcher option (available under dispatcher and service) 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 option may be set either directly at the dispatcher level to establish a default value for all Dispatcher servioces, or may be set for specific named services. The default value is 5.

54.2.16 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 servies, 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.

Prior to 7.0.5, the default value had been 0 while initial configuration set this to 600. As of 7.0.5, the default value is 600 seconds (10 minutes).

54.2.17 max_life_conns Option Under dispatcher

As part of the Dispatcher's ability to perform Worker Process housekeeping, the max_life_conns Dispatcher option (available under dispatcher and service) 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 an SMTP server process 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.

54.2.18 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 max_life_time option may be set either directly under dispatcher, as a default for services, or may be set at the named service level to affect solely that named service. The Dispatcher's default is 1 day (86400).

54.2.19 max_procs Option

The max_procs Dispatcher option (available under dispatcher and service) controls the maximum number of Worker Processes that will be created for a service. Thus this value times max_conns specifies the maximum number of simultaneous connections that can be handled. The max_procs option may be set either directly under dispatcher, as a default for all services, or may be set for a named service to affect solely that named service. 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.

54.2.20 max_shutdown Option

The max_shutdown Dispatcher option (available under dispatcher and service) specifies a maximum number of processes that may be shutting down. In order to provide a minimum availability for the service, the 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. The max_shutdown option may be set either directly under dispatcher, as a default for all services, or may be set for a named service to affect solely that named service. As of 7.0.5, the default value is 5.

54.2.21 min_procs Option

The min_procs Dispatcher option (available under dispatcher and service) determines the minimum number of Worker Processes that will be created by the Dispatcher for the current service. The min_procs option may be set either directly under dispatcher, as a default for all services, or may be set for a named service to affect solely that named 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.

54.2.22 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 using parameter to set a channel name 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.

54.2.23 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.

54.2.24 Service group

Besides a few options affecting Dispatcher operation as a whole, or set directly at the dispatcher level to establish defaults for Dispatcher services lacking their own specific settings, the primary way that Dispatcher options are set is within named service groups, to define specific Dispatcher services. Each such named service has associated ports (established via the tcp_ports and/or ssl_ports options); the Dispatcher listens for TCP connections on those ports, and then hands off the connections to Worker Processes for the named service, where the Worker Process will then execute the image specified for that service.

For instance, at its most basic, in principle the SMTP service *could* be defined as:

```
msconfig> set dispatcher.service:SMTP.tcp_ports 25
msconfig# set dispatcher.service:SMTP.image IMTA_BIN:tcp_smtp_server
msconfig# set dispatcher.service:SMTP.enable 1
```

More typically, however, additional options, such as a logfilename and perhaps parameter as well as various options tuning performance would be set also; while since the enable Dispatcher service option defaults to 1, and as of MS 7.0.5 the named SMTP service defaults to the image shown, in reality neither of those options needs to be *explicitly* set.

Within a named Dispatcher service group, besides the port tcp_ports and/or ssl_ports) and image option settings establishing the correspondence between port(s) and service, many other Dispatcher option settings are available to specify and further control operation of that service: thresholds at which to create new or additional server processes, age at which to expire "old" server processes, *etc*.

Thus in particular note that in MTA terminology and operation, a Dispatcher *service* such as the SMTP service encompasses potentially multiple (transient) SMTP *server processes*: the SMTP server processes come and go, created and expired as needed by the Dispatcher, within the parameters established in the Dispatcher's SMTP service definition.

Note that special symbolic names known to the MTA -- such as IMTA_BIN and IMTA_LOG -- are supported (and convenient) for values of service group options.

54.2.25 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.

54.2.26 ssl_ports Option Under service

The ssl_ports Dispatcher service 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.

Note that channels for servers using ssl_ports need to set maytlsserver or musttlsserver to properly initialize as an SSL server.

See also the tcp_ports Dispatcher service option.

54.2.27 tcp_ports Option Under 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.

54.2.28 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.

54.2.29 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.

54.2.30 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.

54.2.31 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.

54.2.32 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.

54.2.33 loglevel Option Under mta

The mta.logfile.loglevel option is ignored by the Dispatcher and Job Controller unless the dispatcher.use_nslog and job_controller.use_nslog options are set for them, respectively.

Note that this MTA-wide setting of mta.logfile.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.

Also note that the mta.loglevel option is defined and can be set to the same set of values, but as of MS 8.0.1.2 it is a no-op. It's behavior in previous version is erratic - in some cases it works and mta.logfile.loglevel does not, in other cases the value is ignored.

54.2.34 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`

54.3 Dispatcher debugging and log files

Dispatcher error and debugging output (if enabled) are by default written to the file `dispatcher.log` in the MTA log directory. (But see the `use_nslog` Dispatcher option if you prefer to direct Dispatcher error and debugging output to an nslog file.)

Debugging output may be enabled using the `debug` Dispatcher option in Unified Configuration (or the option DEBUG in the Dispatcher configuration file in legacy configuration), or on a per-process level, via the PMDF_DISPATCHER_DEBUG environment variable (UNIX). (In legacy configuration, when enabling the DEBUG option in the Dispatcher configuration file, note that it may be set either globally, or (by setting it within specific SERVICE sections) only for one or more services; in particular, to enable debugging for the Dispatcher but *not* for its worker processes (such as SMTP server processes), it may be set within a [SERVICE=DISPATCHER] section.)

Note that as of MS 7.0.5, enabling the MTA option `debug_flush` will cause Dispatcher debugging to get flushed to disk immediately.

The format of records in the Dispatcher log file is:

hh.mm.ss.ss (four-digit-thread-id): debug-message

Physical lines in the Dispatcher log file are limited to 79 characters. If a record has to be continued to a second line (due to the *debug-message* getting "too long"), then the rest of the *debug-message* will be indented with white space for the *time-stamp* and *four-digit-thread-id* fields, to line up with the initial part of *debug-message*.

Note that the Dispatcher itself always has *thread-id* number 1; that is, records for the main Dispatcher process itself begin

hh.mm.ss.ss (1):

The debug Dispatcher option (DEBUG option in dispatcher.cnf in legacy configuration) or PMDF_DISPATCHER_DEBUG environment variable (UNIX) defines a 32-bit debug mask in hexadecimal. Enabling all debugging is done by setting the option to -1, or by defining the environment variable system-wide to the value FFFFFFFF. The actual meaning of each bit is described in Dispatcher debugging bits.

Table 54.2 Dispatcher debugging bits

Bit	Hexadecimal value	Decimal value	Usage
0	x00001	1	Basic Dispatcher main module debugging
1	x00002	2	Extra Dispatcher main module debugging
2	x00004	4	Dispatcher option (Unified Configuration) or configuration file (legacy configuration) logging
3	x00008	8	Basic Dispatcher miscellaneous debugging
4	x00010	16	Basic service debugging
5	x00020	32	Extra service debugging, including PORT_ACCESS mapping table probes
6	x00040	64	Process related service debugging
7	x00080	128	Shutdown queue debugging
8	x00100	256	Basic Dispatcher and process communication debugging
9	x00200	512	Extra Dispatcher and process communication debugging
10	x00400	1024	Packet level communication debugging
11	x00800	2048	Not used
12	x01000	4096	Basic Worker Process debugging
13	x02000	8192	Extra Worker Process debugging
14	x04000	16384	Additional Worker Process debugging, particularly connection handoffs
15	x08000	32768	Not used
16	x10000	65536	Basic Worker Process to Dispatcher I/O debugging
17	x20000	131072	Extra Worker Process to Dispatcher I/O debugging, including some TLS initialization debugging

20	x100000	1048576	Basic statistics debugging
21	x200000	2097152	Extra statistics debugging
24	x1000000	16777216	Log PORT_ACCESS, dns_verify_domain, and enable_rbl denials to the dispatcher.log file

Chapter 55 Job Controller

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 is 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.

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.

55.1 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. This data structure is referred to as the *queue cache database*. 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 (maxjobs). 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.

55.1.1 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 enqueuing 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.

[1] The effective `threaddepth` never goes over 134,217,727, and the effective job limit never goes below 1.

55.1.2 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 `mtpriority_policy` 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). Priority handling 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 among those 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 *`priority`*`backoff` 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 *`priority`*`notices` 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 *`priority`*`blocklimit` 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.

55.2 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 channnels are read
! indirectly if this include file exists.
!
<IMTA_TABLE:job_controller.site  (4)
```

```
!
! Pool definitions
!
[POOL=DEFAULT]  (5)
JOB_LIMIT=10    (6)
!
[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.

55.3 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.

When the `job_controller.use_nslog` option has been enabled, see also the logfile options set as `job_controller.logfile.*`.

55.3.1 `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.imta.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.

55.3.2 `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 string disambiguifying the file name. (Note that the `imsimta purge` utility understand the *`uniqueid`*s and can be used to purge back older log files.)

55.3.3 `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.

55.3.4 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 (unsigned integer) 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".

55.3.5 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.

55.3.6 max_cache_messages Option

The max_cache_messages option is available only as a global Job Controller option, job_controller.max_cache_messages.

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.

55.3.7 `max_life_askwork` Option

The `max_life_askwork` Job Controller option (available both at global `job_controller` level to set a default for all `channel_class`es, or at `channel_class` level to specify the limit for that specific type of channel) imposes another limit on channel master job life expectancy, in addition to the limit set for each 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.

55.3.8 `max_life_conns` Option Use With max_life_conns Under job controller

For the Job Controller, the legacy configuration `max_life_conns` option has been renamed to `max_life_askwork`. (The preferred name `max_life_askwork` is supported in legacy configuration as of MS 7.0.5.)

55.3.9 `max_life_time` Option Under job controller

The `max_life_time` Job Controller option (available both at global `job_controller` level to set a default for all `channel_class`es, or at `channel_class` level to specify the limit for that specific type of channel) 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.

55.3.10 `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.

55.3.11 port Option Under job controller

The port Job Controller option specifies the TCP port that the Job Controller uses for inter-process communication.

55.3.12 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".

55.3.13 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).

55.3.14 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 not available under job_pool (or in legacy configuration, is ignored inside of a [POOL] section).

55.3.15 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.

55.3.16 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 before the stress level starts being lowered. The default is 120 seconds, *i.e.*, two minutes.

Note that the stress level actually lowered when a new job controller job is initiated. As of the 8.0.2 release, the stress level is also lowered when unstresscount messages have been processed by the channel and stresstime time has elapsed without any indication of stress. unstresscount defaults to 10000.

55.3.17 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.

55.3.18 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.

55.3.19 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.

55.3.20 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.

55.3.21 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 fist 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.

55.3.22 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_class or job_pool groups. The default is 27442.

55.3.23 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.

55.3.24 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`.

55.3.25 loglevel Option Under mta

The `mta.logfile.loglevel` option is ignored by the Dispatcher and Job Controller unless the `dispatcher.use_nslog` and `job_controller.use_nslog` options are set for them, respectively.

Note that this MTA-wide setting of `mta.logfile.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.

Also note that the `mta.loglevel` option is defined and can be set to the same set of values, but as of MS 8.0.1.2 it is a no-op. It's behavior in previous version is erratic - in some cases it works and `mta.logfile.loglevel` does not, in other cases the value is ignored.

55.3.26 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.

55.3.27 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.

55.4 Checking that the Job Controller is running

You may check that the Job Controller is running with the command `imsimta process`. 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).

The Job Controller log file, `job_controller.log-`*`uniqueid`*, may be inspected to check for any Job Controller error messages. Note that, unless Job Controller debugging has been enabled, the Job Controller only writes to its log file if it encounters an error condition; that is, the Job Controller log file will be empty under normal, no-errors, no-debugging-enabled, conditions.

Chapter 56 Compiling the MTA configuration

When using a compiled configuration, whenever you make a change to the MTA configuration, such as to mappings, rewrite rules, channels, aliases, conversions, dispatcher configuration (8.0 or later), job controller configuration (8.0 or later), or channel-specific options (8.0 or later), you must recompile your configuration for the changes to take effect. Compilation reads the entire configuration and writes it out as a single binary file.

Whenever a component of the MTA (such as a channel program) must read the configuration file, it first checks to see if a compiled configuration file exists. If it does, the image is loaded instead of reading the various configuration files.

The command to compile the MTA configuration is:

```
imsimta cnbuild
```

Prior to MS 7.0.5, two MTA Tailor options were relevant for `imsimta cnbuild`: `imta_config_data` specified the default output image file, and `imta_option_file` specified an option file adjusting configuration internal table sizes. As of MS 7.0.5, these MTA Tailor options have been deleted, and hard-coded file paths are used instead, *config-root*`/advanced/config_data` and *server-root*`/lib/option_config.dat`, where *server-root* is the install directory or the value of the `SERVERROOT` environment variable.

Compiled configurations have a static part and a dynamic reloadable part. If the dynamic part is changed, and you run the `imsimta reload` command, all running MTA processes will reload the dynamic data. The dynamic parts are presently the mapping tables and text databases. Everything else is static.

Using a compiled configuration makes it easy to test configuration changes because the configuration files themselves do not "go live" until they are compiled. Most MTA test utilities provide a `-noimage` switch which tells them to ignore the compiled configuration and instead read the configuration from the original files. This lets you test things to make sure they are working before affecting the running system.

If you make changes to your mappings or reverse, forward, or general text databases, then after recompiling issue the command `imsimta reload` to get the changes to take effect. Note that a compiled configuration is a prerequisite to using `imsimta reload`.

Note that `imsimta reload` also reloads authentication, SSL/TLS, and as of Messaging Server 8.0.2, affinity group options. This is done even if the MTA's configuration is not compiled.

If you make changes to other parts of the configuration you must restart the components whose configuration was changed. Some of the dependencies are:

imsimta restart smtp	Changes to mappings, aliases, channels, rewrite rules, channel-specific options for SMTP channels, or text databases.
imsimta restart dispatcher	Changes to the dispatcher configuration, mappings, aliases, channels, rewrite rules, channel-specific options for SMTP channels, text databases

imsimta restart job_controller	Changing the job controller configuration, adding or deleting channels, or changing the values of any of the channel options `pool`, `maxjobs`, `master`, `slave`, `single`, `single_sys`, or `multiple`. Note that adding or changing a `threaddepth` channel option value can be dealt with instead by using the `imsimta cache -change -thread_depth=...` command. You must also restart the job controller if you want changes to master channel jobs to take effect immediately (rather than waiting for the controller to time-out existing channel jobs).

Try to avoid restarting the Job Controller, especially at times when large numbers of messages are in the queues, unless one of the preceding conditions necessitates a restart.

The following changes for compiled MTA configurations were introduced in Messaging Server 8.0:

- The dispatcher configuration, job controller configuration, and channel-specific options are now part of the compiled configuration.
- The `configure` command no longer generates a compiled configuration by default, for both legacy and Unified Configuration.
- Unified Configuration provides the `-xmlfile=xml-config-file` switch in various test utilities like `imsimta test -rewrite`. When combined with the `-noimage`, this allows testing of configuration files in non-default locations.
- The `imsimta version` command now shows if a compiled configuration is used or not.

Chapter 57 Mail filtering and access control

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 and LMTP connection attempts based on source IP address; 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.

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. 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`
  - `AUTH_DEACCESS`
  - `MX_ACCESS`
  - `IP_ACCESS`

57.1 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.

57.1.1 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 the mapping was present when the Dispatcher was started, then Dispatcher operation will include checking the mapping for each incoming connection. For each incoming connection 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.

As of MS 8.0.1.3, tildes (~) may be used as delimiters between multiplie lines in order to create a multiline response. Also 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 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 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 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 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 MS 6.3-0.15, such SMTP server probes of `PORT_ACCESS` will respect the `$N` (in the case of SMTP AUTH usage), `$>`, and `$<` flags, whereas in prior versions the SMTP server probe results were only relevant for setting the SASL ruleset and the optional logging text; as of the fix for 12208860 (Sun 6590888) (MS 6.3-5.02), `$N` rejections will be respected in all cases. Thus new in 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 57.1 `PORT_ACCESS` mapping flags

Flag	Description
`$U`	(New in 6.3-0.15) Enable channel debugging. As of 7.3-11.01, this includes consulting the `mm_debug` and `os_debug` MTA options and enabling any debugging they specify. This is only supported for SMTP server probes of the `PORT_ACCESS` mapping table; it is not supported for Dispatcher probes of the `PORT_ACCESS` mapping table.
`$G`	(New in MS 7.0u5 for SMTP server; new in MS 8.1 for LMTP server) Enable `TRACE_LEVEL=2` channel debug output. Only supported for SMTP or LMTP server probes; not supported for Dispatcher probes.
`$V`	(New in MS 7.0) Enable the MTA's private SMTP extensions XADR, XCIR, XGEN, and XSTA, overriding any SMTP server `DISABLE_*` TCP/IP-channel-specific options. Only supported for SMTP server probes; not supported for LMTP server or Dispatcher probes.
`$/`	(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 SO_LINGER enabled and a timeout of 0, which may clear slots quicker on intermediate firewalls and proxies. Only supported for SMTP server and Dispatcher probes; not supported for LMTP probes.

$V	(New in 7.0-0.04) Enable the MTA's private SMTP extensions XADR, XCIR, XGEN, and XSTA, overriding any SMTP server DISABLE_* TCP/IP-channel-specific options. 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.
$Y	Allow access.
$T	If bit 1 of the log_connection MTA option is set, and if a connection is rejected ($N is also specified), then write a connection log file "T" record, including any of the optional text 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[1]
$A*string*	(New in Messaging Server 7.4-18.01; for LMTP server, new in MS 8.0.1.) 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 and LMTP 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.
$N*string*	Reject access with the optional error text *string*.
$F*string*	Synonym for $N*string*, *i.e.*, reject access with the optional error text *string*.
$D*delay*	(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 LMTP server or Dispatcher probes of the PORT_ACCESS mapping table.
$S*channel-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 LMTP server or Dispatcher probes of the PORT_ACCESS mapping table.
Additional non-flagged fields	Description
TLS-certificate-nicknames	(New in Messaging Server 7.0-0.04) Comma-separated list of TLS certificate nicknames (which must appear subsequent to a vertical bar character). This is only supported for SMTP server probes of the PORT_ACCESS mapping table; not supported for LMTP server or Dispatcher probes of the PORT_ACCESS mapping table.
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.

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

[1] To use multiple flags with arguments, or the non-flagged fields, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

Note that prior to 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 Dispatche - or as of MS 6.3-0.15 use the imsimta 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. (Note that this requires a compiled MTA configuration and you'll first need to recompile before reloading.)

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.

57.1.1.1 Initial PORT_ACCESS mapping table

Initial configuration will generate a basic PORT_ACCESS mapping table that makes use of a subsidiary INTERNAL_IP mapping table to recognize "internal" *vs*. "external" IP sources:

```
PORT_ACCESS

  *|*|*|*|*  $C$|INTERNAL_IP;$3|$Y$E
  *  $YEXTERNAL
```

57.1.2 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").

57.1.3 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 additional, 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_spares1 (which in 8.0.2.2 replaced the previous include_spares MTA option) 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-chan|*from-addr*|*dst-chan*|*to-addr*|*orcpt*|*access-counts*|*list-of-tags*|*s1*|*s2*|*s3*|*s4*|*s5*|*s6*

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/TLS-*crypto-info*/*claimed-HELO-name* 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_spares1 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_spares1 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_spares1 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.) The specific enhanced status code, which in turn determines whether the error is temporary or permanent, can be specified through the use of $X; see the access mapping flags table below for details.

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, 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 specifing 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 57.2 Address access mapping table flags[1]

Output flag	Description
$/	(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 SO_LINGER enabled and a timeout of 0, which may clear slots quicker on intermediate firewalls and proxies.[3]
$!	Disable (Sieve requested) vacation messages regarding this message[3]
$*	(New in MS 7.0 for FROM_ACCESS; new in MS 7.0u2 for the other address *_ACCESS mappings) If used with $N, force disconnect of the SMTP session.
$+L	(New in MS 7.0.5) If used with $M in FROM_ACCESS, cause the captured message copy to be in journal format
$B	Redirect the message to the bitbucket
$H	Hold the message as a .HELD file
$J	(New in MS 7.0u4) If used with $M, 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 $.text. 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 *all* recipients of this message copy.
$v	(New in MS 7.0u3) Force Sieve filter discard behavior for solely this recipient[5]. For FROM_ACCESS, $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 *all* 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 FROM_ACCESS,

	$z remains synonymous with $Z as meaning force Sieve filter jettison for all recipients.
Output flags with arguments, in argument reading order[2]	
$U*n*	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 *channel-name* -- 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 *changes* the source channel, then the FROM_ACCESS check process is restarted; also, $~ can only be applied once.[3]
$J*address*	Replace original envelope From address with specified address[3]
$K*address*	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, *on its own*, cause addition of a Sender: header line showing the new sender address.)
$I*user|identifier*	Check specified user for specified groupid (UNIX) and if not in the group, reject access (effectively sets the $N output flag).
$<*string*	Send *string* to syslog (UNIX) if probe matches; see also the sndopr_priority MTA options[4]
$>*string*	Send *string* to syslog (UNIX) if access is rejected; see also the sndopr_priority MTA option[4]
$D*delay*	Delay response for an interval of *delay* 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).
$T*probe-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.
$A*header*	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).
$G*conv-tag*	(New in MS 6.0) Add the conversion tag (or comma-separated tags) *conv-tag* to the message.
$M*address*	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.
$S*x* $S*x*,*y* $S*x*,*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]
`$,`*x*	(New in 6.1) Set a spam level value *x* (between -10000 and 10000). If the message already had a spam level, this is a `"spamadjust"` effect (adds or subtracts the specified amount *x* from the prior spam level). Note that such a spam level/spamadjust effect applies to all recipients (even for the recipient-specific mapping tables such as `SEND_ACCESS`) in order for tests to see if one of the recipients is a "honeypot" address.
`$Q`*language* `$Q`*language*`|`*country*	(New in MS 6.3) Set a preferred language and optionally a preferred country[3]
`$(`*postmaster-address*	`FROM_ACCESS`[6]: (New in MS 6.3) Set an override postmaster address[3]
`$)`*postmaster-address*	(New in MS 6.3) Set an override postmaster address if none was previously set[3]
`$+E`*name*`|`*value*	(New in MS 7.0u2) Set the Sieve environment item *name* to *value*
`$+R`*n*`|`*string*	(New in MS 7.0u2) `FROM_ACCESS`[6]: Opt-in to spam/virus filtering.[7] *n* specifies which spam/virus filter package; *string* is the opt-in string to pass to that spam/virus filter package.[3] The string may be blank but the preceding vertical bar may not be omitted.
`$+R`*n1*`,`*s1*`,`*n2*`,`*s2*`,...`	(New in MS 8.0.2.3) `FROM_ACCESS`[6]: Opt-in to spam/virus filtering.[7] *n1, n2 ...* specifies the spam/virus filter packagew; *s1, s2, ...* specify the opt-in strings to pass to the associated spam/virus filter package.[3] The opt-in strings may be blank but the commas may not be omitted.
`$+^`*n1*`,`*n2*`,...`	(New in MS 8.0.1.3) `FROM_ACCESS`: Disable selected filters, overriding any source channel optins. The argument is a comma-separated list of integer values *n1, n2, n3 ...* specifying which spam filters to disable.
`$+&`*n1*`,`*s1*`,`*n2*`,`*s2*`...`	(New in MS 8.0.2.2) `FROM_ACCESS`: Load sender spare attribute with specified value The argument is a comma-separated list of integer-string value pairs *n1, s1, n2, s2 ...* specifying the index of the attribute to load and the value to load into it.
`$W`*n*	(New in MS 8.0) `FROM_ACCESS`: Set the MT-PRIORITY level to *n*.
`$X`*error-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
`$N`*string*	Reject access with the optional error text *string*
`$F`*string*	Synonym for `$N`*string*, *i.e.*, reject access with the optional error text *string*
`$(`*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

	`error_text_*` error text option values with values from the file *`file-spec`*; the values should be specified one per line in the file, in the order of the `error_text_*` options as shown in `error_text_*` MTA options.[5]
`$S`*`url`*	`[ORIG_][SEND|MAIL]_ACCESS`[6]: Apply the Sieve filter obtained from resolving *`url`*[5]
`$+R`*`n`*`|`*`string`*	(New in 7.0u2) `[ORIG_][SEND|MAIL]_ACCESS`[6]: Opt-in to spam filtering.[7] Only applied if neither `$N` nor `$F` is set. *`n`* is which spam/virus filter package; *`string`* is the opt-in string to pass to that spam/virus filter package.[5] The string may be blank but the preceding vertical bar may not be omitted.
`$+R`*`n1,s1,n2,s2,...`*	(New in MS 8.0.2.3) `[ORIG_][SEND|MAIL]_ACCESS`[6]: Opt-in to spam filtering.[7] Only applied if neither `$N` nor `$F` is set. *n1, n2, ...* specify which the spam/virus filter packages to activate; *s1, s2, ...* specify the the opt-in strings to pass to the associated spam/virus filter package.[5] The opt-in strings may be blank but the commas may not be omitted.
`$+^`*`n1,n2...`*	(New in MS 8.0.1.3) `[ORIG_][SEND|MAIL]_ACCESS`[6]: Disable selected filters, overriding any active optins. The argument is a comma-separated list of integer values *n1, n2 ...* specifying which spam filters to disable. Note that this effect only extends to the current level of alias expansion - optins at inner levels will be honored.
Input flag comparisons	Description
`$:|`	(New in MS 7.0) Match only if external material (*e.g.*, an envelope address) in the probe contained a vertical bar
`$;|`	(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 *not* been used
`$:D`	Match only if DELAY delivery receipts have been requested (*e.g.*, NOTIFY=DELAY)[5]
`$;D`	Match only if DELAY delivery receipts have *not* 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
`$:F`	Match only if FAILURE delivery receipts have been requested (*e.g.*, NOTIFY=FAILURE)[5]
`$;F`	Match only if FAILURE delivery receipts have *not* 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 (*e.g.*, NOTIFY=SUCCESS)[5]
$;S	Match only if SUCCESS delivery receipts have *not* been requested[5]
$:T	Match only if TLS has been used
$;T	Match only if TLS has *not* 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]
$:C	(New in MS 8.0.1.3) Match only if message is the result of a capture/journal action.
$;C	(New in MS 8.0.1.3) Match only if message is not the result of a capture/journal action.

[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.

[7] Only one $+R and corresponding value can be used in a single mapping result.

+ 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.

An example of an important use of the ORIG_SEND_ACCESS mapping table is discussed in Blocking SMTP relaying.

57.1.3.1 Initial SEND_ACCESS mapping table

Initial configuration will generate a basic SEND_ACCESS mapping table to immediately issue errors in cases of attempts to submit messages to certain invalid domain names:

SEND_ACCESS

```
tcp_*|*|*|*@[127.*]  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@localhost.*  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@example.com  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@example.net  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@example.org  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@*.test  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@*.example  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@*.invalid  $X5.1.2|$NBad$ destination$ system
tcp_*|*|*|*@*.localhost  $X5.1.2|$NBad$ destination$ system
```

57.1.4 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 SMTP/*claimed-HELO-name* in the case of messages submitted via SMTP, or SMTP/TLS-*crypto-info*/*claimed-HELO-name* 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_spares1 (renamed from include_spares as of MS 8.0.2.2) and access_auth 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|app-info|submit-type|src-chan|from-addr|auth-from|source-auth|username|list-of-tags|s1|s2|s3|s4|s5|s6

The *source-auth* field contains the value of the SMTP MAIL FROM command's AUTH parameter; the *username* field contains the canonical username result produced by authentication. 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 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.

This configuration can easily be extended to block submission entirely based on various criteria. For example, suppose that in addition to not being allowed to receive mail, is is also desirable to prevent over quota users from sending mail. The following settings would expose the mailUserStatus attribute in the sixth spare slot:

```
msconfig> set ldap_spare_6 mailUserStatus
msconfig# set include_spares1 32
```

And the following addition at the end of the FROM_ACCESS mapping would check the attributes value:

```
FROM_ACCESS

  TCP|*|SMTP*|MAIL|tcp_*|*|*|overquota $X4.2.3|$NOver$ quota$ users$ cannot$ send$ mail
```

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.

57.1.4.1 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
```

57.1.5 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.

For incoming SMTP messages, the Dispatcher consults the PORT_ACCESS mapping table to decide whether or not to accept the SMTP 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 (incoming) 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 MS 6.2) 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 for incoming messages 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),
- SASL_ACCESS (after SMTP AUTH),
- 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.*).

The MTA is designed to do most processing upon message enqueue. So outgoing SMTP messages receive relatively little processing. However, there are a few possible access checks for outgoing SMTP messages. See the `AUTH_ACCESS`, `IP_ACCESS`, and `TLS_ACCESS` mapping tables.

57.2 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.

The MTA is designed to continue operating even when impacted by an attempted denial of service attack. The design of the Dispatcher includes limits, which are configurable, on how many simultaneous inbound connections to accept, how many processes to generate to handle inbound connections, and how to apportion such inbound connections to those Worker Processes. Similarly, the Job Controller has various self-managing features; see the discussion of Job Controller operation under stress. However, sites subject to malicious attacks, or sites that communicate with particularly ill-behaved correspondents, may want to take further explicit steps for protection.

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 `pmxbl` 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_RECIPENTS_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 threshholds 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 58 Spam and virus filtering

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.

When use of multiple spam/virus filter package "slots" is configured, the MTA calls all the configured spam/virus filter packages as messages are being received (enqueued). Thus the spam/virus filter package processing of messages is essentially in parallel with each other (and with the MTA's own inherent processing). The MTA then uses (applies) the spam/virus filter package results -- their respective verdicts -- in the order of the configured spam/virus filter package "slots": `spamfilter1*`, before `spamfilter2*`, before `spamfilter3*`, *etc.*

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 Mail 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;
- the call-out approach permits the use of multiple spam/virus filter packages "in parallel", which for sites that like to use (or compare) different spam/virus packages has benefits including:

  1. higher throughput due to the parallelization of the MTA calling all the spam/virus filter packages in parallel while messages are being enqueued,
  2. convenient comparison of and swapping between different spam/virus filter packages at site convenience,
  3. the potential for complex combining and weighting of verdicts from different spam/virus filter packages,
  4. provisioning of user use of email spam/virus services that is integrated with the provisioning of user email addresses, routing, and other email services.

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.

58.1 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 file 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 58.1 Some Brightmail configuration file options

Option name	Meaning	Typical setting for MTA use
blSWPrecedence	A message may receive multiple forms of processing hence multiple verdicts; for instance, receive verdicts of both `spam` and `virus`. This option specifies the order in which verdicts are processed for messages with multiple verdicts. Verdicts should be separated with the hyphen character, `-`. A verdict appearing first (left-most) in the hyphen-separated list will be processed before a verdict appearing next (to the right), *etc.*. Supported verdicts are `virus` and `spam`. Brightmail recommends that virus verdicts be processed first.	`blSWPrecedence: virus-spam`
blSWClientDestinationDefault	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 `inbox` has this meaning, and hence is usually the recommended setting.	`blSWClientDestinationDefault: inbox`

blSWLocalDomain	Occurrences of this option specify which domain(s) are considered local, hence which domains get the handling specified by blSWClientDestinationLocal options (as opposed to handling specified by blSWClientDestinationForeign options).	`blSWLocalDomain:`
		`  *your-domain-1*.com`
		`blSWLocalDomain:`
		`  *your-domain-2*.com`
		`...`
		`blSWLocalDomain:`
		`  *your-domain-n*.com`

blSWClientDestinationLocal	This option specifies *verdict-name*|*destination-data* pairs for recipients in a local domain (a domain specified by a `blSWLocalDomain` option). (`blSWClientDestinationForeign` 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 `spamfilterN_null_action=data,discard;` MTA option setting) to consider a null *destination-data* 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:	The default settings are:
		
		`blSWClientDestinationLocal:`
		`  spam|junkmail`
		`blSWClientDestinationLocal:`
		`  virus|`
		
		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.

	spamfilter*N*_string_action=data:,require "fileinto"; fileinto "$U"; This typically but not necessarily amounts to *verdict-name*|*folder-name* pairs, with a null (empty) folder-name meaning to discard the message. But more precisely, it truly consists of *Brightmail-verdict-name*|*MTA-verdict-string* pairs, where the *MTA-verdict-string* can be specified in an MTA option of the form spamfilter*N*_verdict_M=*MTA-verdict-string* to correspond to an action specified in a spamfilter*N*_action_m option, and with a null *MTA-verdict-string* being handled in accordance with the MTA's corresponding spamfilter*N*_null_action option. Supported *verdict-name* values include spam and virus.	
blSWClientDestinationForeign	Analogous in syntax and meaning to blSWClientDestinationLocal settings above, but applying to recipients who are not in one of the blSWLocalDomain domains.	
blSWClientOptin	This option specifies the handling of opt-in decisions by the Brightmail Client, and for use with the MTA must be set to TRUE.	blSWClientOptin: TRUE
blswcServerAddress	This option specifies the IP address(es) and port(s) of one or more Brightmail servers. The syntax is *ip*:*port*[,*ip*:*port*,...]	blswcServerAddress: *ip*:*port* or blswcServerAddress: *ip-1*:*port-1*, *ip-2*:*port-2*,..., *ip-n*:*port-n*
blCommonDebugFilename	This option specifies the location of Brightmail's own debug log file. It can be set to any of syslog or syslog|syslog-facility (the default when logging to syslog if the facility is not set is mail), stderr, or a *path-to-filename*.	blswcCommonDebugFileName: /opt/mailwall/logs/common_log
blCommonDebugLevel	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 ALL, 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 ALL,6. And Brightmail recommends that after solving the difficulty, that the option be set back to ALL,4.	blCommonDebugLevel: ALL,4
blswcDebugFileName		blswcDebugFileName: /opt/mailwall/logs/bmclient_log
blswcDebugLevel		
blswsDebugFileName		blswsDebugFileName: /opt/mailwall/logs/bmserver_log
blswsDebugLevel		

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.

58.2 ClamAV spamfilterN_config_file

When using ClamAV, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter3_config_file IMTA_TABLE:clamav.dat
```

With ClamAV, the file that a `spamfilterN_config_file` MTA 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).

58.3 ICAP spamfilterN_config_file

When using ICAP, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter4_config_file IMTA_TABLE:icap.dat
```

With ICAP, the file that a `spamfilterN_config_file` MTA 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)

58.4 Milter spamfilterN_config_file

When using Milter, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter4_config_file IMTA_TABLE:milter.dat
```

With milter (supported as of MS 6.3), the file that a `spamfilterN_config_file` MTA option names may contain the following options:

- `CONNECT_TIMEOUT` (integer). (New in 8.0) This option provides a separate timeout for the initial milter connection separate from the timeout waiting for milter responses. 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 responseand 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). (New in MS 8.0.1) 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.
- `HOST` (hostname or IP address). Specify a host running a Milter server. A value must be specified for this option; (its presence is required).

- DATA_IN_BODYEOB (0 or 1; default is 0). (New in 8.0.2, and for libmilters.so new in a patch to MS 7.0.5) When set to 0 (the default), message body material is not sent as part of the milter BODYEOB (body end of body) command. Setting this option to 1 allows message body material to be sent with the BODYEOB command, which while legal per the milter specifications and more efficient, may cause trouble with milters such as Proofpoint's milter.

- IGNORE_BAD_CERT (0 or 1; default is 0). (New in 8.0.1.3.) Setting this option to 1 disables SSL/TLS certificate checking. This option is only meaningful if the USE_SSL option is set to 1.

- IMMEDIATE_HEADER_MODIFICATIONS (integer, default 0). (New in MS 8.0.1) 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. Additionally, the 8.0.1 release of this capability implements different semantics for multiple deletes with different indices than would be obtained otherwise. These semantics have been brought in line with normal milter operation as of MS 8.0.2.2.

- 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.

- 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.

- PORT (integer). Specify the port on which the Milter server is listening. A value must be specified for this option; (its presence is required).

- 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 (SMFIF_QUARANTINE) 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;".

- REPROCESS_CONNECT_TIMEOUT (integer). (New in 8.0.2.3) This option's value is used instead of the CONNECT_TIMEOUT value when a message is undergoing reprocessing. This allows a longer timeout to be used in the case where there's no protocol session and thus no need for quick completion. If this option is not set, it defaults to the value set for the REPROCESS_TIMEOUT option. A non-positive setting for REPROCESS_CONNECT_TIMEOUT, whether explicitly set or inherited from TIMEOUT, will result in using a REPROCES_CONNECT_TIMEOUT value of 60.

- REPROCESS_TIMEOUT (integer). (New in 8.0.2.3) This option's value is used instead of the TIMEOUT value when a message is undergoing reprocessing. This allows a longer timeout to be used in the case where there's no protocol session and thus no need for quick completion. If this option is not set, it defaults to the value set for the TIMEOUT option. A non-positive

setting for REPROCESS_TIMEOUT, whether explicitly set or inherited from TIMEOUT, will result in using a REPROCES_TIMEOUT value of 240.

- 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 a FROM_ACCESS and recipient *_ACCESS mapping table.)
- TIMEOUT (integer; default is 3600). Attempting to set a non-positive value will result in a value of 120 being used.
- 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.
- TCP_NODELAY (0 or 1; default 0). (New in 8.0.1.3.) Setting this option to 1 causes the NODELAY flag to be set at the TCP level on all milter connections. Note that the behavior of the milter protocol is highly dependent on what options are negotiated: One milter may require many round trips per message while another may only need one. As such, it isn't clear that there's an optimal setting for the NODELAY flag.
- 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).
- USE_SSL (0 or 1; default is 0). (New in 8.0.1.3.) Setting this option to 1 enables use of SSL/TLS on the milter connection. Note that libmilter.so does not provide support for SSL/TLS so a proxy/tunnel server such as stunnel must be placed in front of most milters before this option can be used.

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 spamfilter*N*_*_action options are spamfilter*N*_null_action (which has a proper default value) and spamfilter*N*_string_action; the spamfilter*N*_action_*M* and spamfilter*N*_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 spamfilter*N*_string_action MTA option (to its special-for-Milters value of data:,$M) as the default value for spamfilter*N*_string_action is not appropriate for Milter use.

58.5 SpamAssassin spamfilterN_config_file

When using SpamAssassin, a `spamfilterN_config_file` MTA option might be set as

```
msconfig> set mta.spamfilter1_config_file IMTA_TABLE:spamassassin.dat
```

With SpamAssassin (supported as of MS 6.1), the file that this option names may contain the following options:

- DEBUG (0 or 1 up through MS 6.2; in 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 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.
- USE_CHECK (integer), default 1. If set to 1, use the spamd CHECK command rather than SYMBOLS command
- (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 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 MS 6.2p1) 3 are supported.)
- FIELD default "Spam-Test".
- VERDICT
- USE_CHECK (0 or 1), default 1.
- USERNAME
- (New in 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 preceeding 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
```

58.6 Archive spamfilterN_config_file

As of 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 spamfilter*N*_config_file MTA option might be set as

```
msconfig> set mta.spamfilter7_config_file IMTA_TABLE:archive.dat
```

With AXS:One, it.com or Microsoft Exchange Journal format use, the following options are available; see MTA configuration for AXS:One 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.

58.7 Sieve spamfilterN_config_file

As of MS 8.0.2, the MTA provides a simple spamfilter plugin, libsieve.so, that can be used to execute a fixed Sieve script at a particular point in the spam filter evaluation/interpretation process. The primary purpose of this plugin is for testing, but it can be used in production environments to execute a "fixup" script between two other spam filters.

The configuration file for this plugin consists of zero or more options, zero or more blank lines, and then the actual Sieve script. The following options are available:

- DEBUG (integer; default 0)
- RESETDEBUG (integer; default 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 Sieve plugin; if channel debugging is disabled, then debugging is disabled.

58.8 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 routine, `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.)

58.9 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 58.2 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 58.3 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 58.4 Available milter macros

Name	Protocol Phase	Content
${auth_authen}	MAIL FROM	Authenticated sender address.
${auth_author}	MAIL FROM	The value of the AUTH parameter to MAIL FROM. Added in 8.0.
${client_addr}	CONNECT	The IP address of the SMTP client, expressed as a dotted quad value. Only set when SMTP over TCP is being used.
${destination_channel}	RCPT TO	MTA destination channel for the current recipient.
$i	MAIL FROM	Queue id for the current message. The MTA generates a unique id for each session; this id is what appears in the $i macro.
$j	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.
${mail_addr}	MAIL FROM	The MAIL FROM address for the current transaction.
${mail_host}	MAIL FROM	The host part of the MAIL FROM address for the current transaction.
${optin}	RCPT TO	Spamfilter optin value for the current RCPT TO adress.
${rcpt_addr}	RCPT TO	Current RCPT TO address.
${rcpt_host}	RCPT TO	The host part of the current RCPT TO address.
${rcpt_mailer}	RCPT TO	Set to "local" for valid recipient addresses, "error" for invalid addresses. New in 8.0.
${source_channel}	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 `spamfilter`*`N`*`_verdict_`*`M`* and `spamfilter`*`N`*`_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 `spamfilter`*`N`*`_null_action` and `spamfilter`*`N`*`_string_action` MTA options are relevant for the MTA's milter configuration.

The default value for `spamfilter`*`N`*`_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 `spamfilter`*`N`*`_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).

58.9.1 MILTER_ACTIONS mapping table

As of MS 8.0.2, the MILTER_ACTIONS can be used to augment or modify the behavior various milter actions. The basic probe format for this mapping is:

tag|number|action

which may then be followed by zero or more additional arguments depending on the action. The basic arguments are:

tag — A tag value which is passed between successive calls to the MILTER_ACTION mapping. The tag value is initially empty.

number — The number of the spam filter slot of the milter issuing the action.

action — The milter action that invoked the mapping.

At present the following milter actions call the MILTER_ACTIONS mapping:

Table 58.5 MILTER_ACTIONS mapping actions and arguments

Action	Additional Argument #1
ACCEPT	
DISCARD	
REJECT	
TEMPFAIL	
REPLYCODE	Milter supplied error message
INSHEADER	Header index, Header name, Header value
ADDHEADER	Header name, Header value

ADDHEADER and INSHEADER support was added in MS 8.0.2.3.

The mapping result consists of metacharacters as well as a series of result strings separated by vertical bars. These strings are consumed by the various metacharacters in the order given below.

$A	Force the action action to ACCEPT.
$T	Set the milter tag to the specified string.
$+^	Disable processing of subsequent spam filters. The specified string consists of a comma-separated list of spam filter index numbers. The specified filters with index values greater than the current filter are shut down and any results they have produced are discarded.
$M (REJECT, TEMPFAIL, and REPLYCODE actions)	Override the error message with the specified string.
$M (ADDHEADER action)	Override header name and/or value. One or two arguments may be supplied separated by a vertical bar. If a single argument is specified it replaces the header value. If two

	arguments are specified the first replaces the header name and the second the header value. If the name is left blank it remain unchanged.
$M (INSHEADER action)	Override header index, name and/or value. One to three arguments may be supplied separated by vertical bars. If a single argument is specified it replaces the header value. If two arguments are specified the first replaces the header name and the second the header value. If three arguments are specified the first specifies a new index, the second a new header name, and the third a new header value. If the name is left blank it remain unchanged.

58.9.2 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_paranoia` 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).

58.9.3 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_specrcpt __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
**              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, SMFIR_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.

58.9.4 Milter errors

A number of milter error conditions can get reported to the MTA, which in turn will generate an SMTP error during the SMTP transaction.

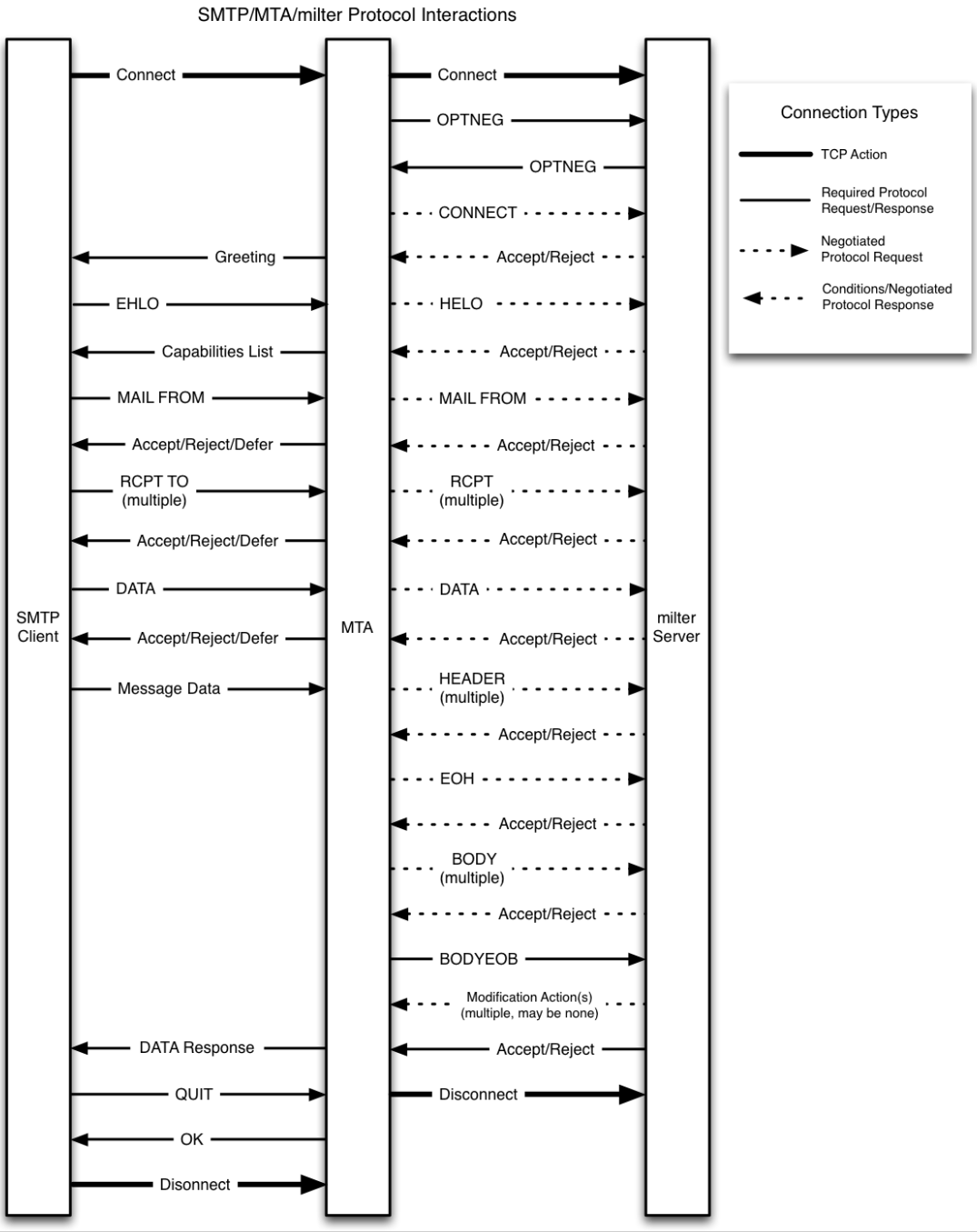
58.10 Milter operation

Milter is implemented as a client-server protocol, with Messaging Server in client role and any of a wide variety of software packages in the server role. Most, but not all, milter servers are written on top of the libmilter library that is provided as part of sendmail.

Unfortunately, the milter protocol provides no authentication, privacy, or integrity protection facilities. As such, it is essential that milter protocol connections be restricted to protected networks, and under no circumstances should they be operated over the open Internet.

The libmilter library is multithreaded and supports a limited amount of threading configuration. Even so, multiple milter servers are often required in large configurations. The Messaging Server milter client does not provide any load balancing capabilities beyond those provided by the DNS (i.e., multiple A/AAAA records), so the use of a load balancer may be required in large, high performance milter configurations.

The milter protocol allows for interaction at every stage of the SMTP protocol. The following diagram illustrates the relationship between a typical SMTP/SUBMIT dialogue and milter protocol commands:



Note that this simplified diagram omits:

- Negotiation of SMTP transport security,
- SUBMIT authentication,
- optional milter macro commands (Macros can precede CONNECT, HELO, MAIL, and RCPT),
- multiple message transfers in the same SMTP/milter session (not used by the MS MTA),

- use of the QUIT_NC command to process multiple connections in the same milter session,
- interactions between multiple milters operating in parallel,
- due to the ability to activate milters based on source or destination channel, milter connect may be deferred to as late as the RCPT TO step in the SMTP diaglogue, and
- if enabled, unknown commands sent by the SMTP will cause the MTA to send UNKNOWN milter commands to the milter server independent of protocol state, and
- PROGRESS responses may be sent by the milter server at any time. These are ignored by the MTA.

The milter Accept/Reject actions are:

- SMFIR_ACCEPT - Accept message unconditionally. No further milter commands or responses will be exchanged.
- SMFIR_CONTINUE - Accept command and continue processing
- SMFIR_DISCARD - Tells the MTA to discard the message. No further milter commands or responses will be exchanged.
- SMFIR_REJECT - In response to a RCPT command, indicates that the recipient should be rejected with a permanent error. In any other context this indicates that the entire message should be rejected with a permanent error and that no further milter commands or responses will be exchanged.
- SMFIR_TEMPFAIL - In response to a RCPT command, indicates that the recipient should be rejected with a temporary error. In any other context this indicates that the entire message should be rejected with a temporary error and that no further milter commands or responses will be exchanged.
- SMFIR_REPLYCODE - In response to a RCPT command, indicates that the recipient should be rejected with the specified error. In any other context this indicates that the entire message should be rejected with the specified error and that no further milter commands or responses will be exchanged.

58.11 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`, `spamtest`, `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", "spamtest"];
spamtest :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.

See the Scheduler's `expire` task for configuration of automatic scheduling of executions of `imexpire`.

Chapter 59 MeterMaid

MeterMaid is a facility that comprises a server, which 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 and MeterMaid client 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.

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. (Note that a named remote_table group may only be set under metermaid_client; a named local_table group may only be set under metermaid.)

Then there are also a number of MTA options that override, for MTA purposes, some of the normal metermaid or metermaid_client options. There are also the viametermaid and metermaidtable IMAP options (under the pwexpirealert group), to specify whether to use MeterMaid, and what MeterMaid table to use, for password expiration alerts.

See also the logfile options, as they are settable under metermaid.

59.1 metermaid options

A MeterMaid server is configured via options under the metermaid group. (Named metermaid groups may be used if desired, but usually are not needed, with options relating to general operation of this MeterMaid server being set directly under metermaid.) Then specific MeterMaid tables maintained by this MeterMaid server are configured under named local_table groups under metermaid (metermaid.table.* configutil parameters in legacy configuration). For instance, on a host running a MeterMaid server with default values for most operational settings, and with one table defined, (bad_password_attempts, perhaps similar to the example Syslog notices after SMTP AUTH attempts with bad password discussed under Triggering effects from transaction logging with LOG_ACTION), one might see:

```
msconfig> show metermaid.*
role.metermaid.enable = 1
role.metermaid.local_table:bad_password_attempts.data_type = string
role.metermaid.local_table:bad_password_attempts.max_entries = 1000
role.metermaid.local_table:bad_password_attempts.quota = 2
role.metermaid.local_table:bad_password_attempts.quota_time = 3600
role.metermaid.secret (suppressed)
```

(Note that use of MeterMaid tables by MeterMaid clients, possibly clients on other hosts, would be configured via settings under metermaid_client on the client host, and especially, settings under a named metermaid_client.remote_table group, or in legacy configuration, metermaid.mtaclient.* configutil parameters on the client host.)

logfile options may also be set under metermaid.

59.1.1 enable Option Under metermaid

The enable MeterMaid option enables the MeterMaid service on start-msg startup.

59.1.2 async Option

The async MeterMaid option sets whether MeterMaid should use asynchronous thread scheduling (default) or the new, experimental linear thread scheduling.

59.1.3 backlog Option Under metermaid

The backlog MeterMaid option specifies the number of connections to permit to be established in the TCP listen queue.

59.1.4 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.

59.1.5 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
```

59.1.5.1 data_type Option

The data_type MeterMaid local_table option specifies the type of data to be stored in this table: one of ipv4, ipv6, or string.

59.1.5.2 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.

59.1.5.3 resubmit_time Option

The resubmit_time MeterMaid local_table 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.

59.1.5.4 inactivity_time Option

The inactivity_time MeterMaid local_table 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.

59.1.5.5 max_entries Option

The max_entries MeterMaid local_table option specifies the maximum number of entries to maintain in this table.

59.1.5.6 table_options Option

The table_options MeterMaid local_table option specifies a space-separated list of options for this table. The currently available options are nocase and penalize.

nocase — When working with data, all comparisons are done using a case-insensitive comparison function. (This option is valid only for string data.)

penalize — After quota_time seconds, the throttle will normally reset the connection count to 0, but if the penalize option is enabled, the throttle will decrement the connection count by quota (but not so the value ends up less than 0) so that additional connection attempts will penalize future quota_time periods. For example, if quota were 5 with a quota_time of 60, and the system received 12 connection attempts during the first minute, the first 5 connections would be accepted and the remaining 7 would be declined. After 60 seconds has passed, the number of connections counted against the particular address would be reduced to 7, still keeping it above quota and declining connection attempts. Assuming no additional connection attempts were made, after another 60 second period, the number of connections would be further reduced down to 2, and MeterMaid would permit connection attempts again.

59.1.5.7 quota Option

The quota MeterMaid local_table option specifies the number of connections to permit per quota_time time period.

59.1.5.8 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.

59.1.5.9 storage Option

The storage MeterMaid local_table option specifies the method of data storage for this table: one of hash or splay.

59.1.5.10 table_type Option

The table_type MeterMaid local_table option specifies the type of table; valid selections include throttle, simple, or greylisting.

59.1.5.11 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. For instance:

```
msconfig> show metermaid.local_table.table-name.*
metermaid.local_table.table-name.table_type = simple
metermaid.local_table.table-name.value_type = string
```

59.1.6 maxthreads Option Under metermaid

The maxthreads MeterMaid option specifies the maximum number of work threads.

59.1.7 port Option Under metermaid

The port MeterMaid option specifies the TCP port number on which MeterMaid listens for connections. The default is 63837.

59.1.8 secret Option Under metermaid

The secret MeterMaid option specifies the secret used to authenticate MeterMaid clients with the server.

59.1.9 sslcachesize Option Under metermaid

The sslcachesize MeterMaid option specifies the number of SSL sessions to be cached by the MeterMaid server. If this is set to 0 or not set, this will use a default provided by the Mozilla NSS library which was 10000 last time this was checked (March 2016).

59.1.10 sslusessl Option Under metermaid

Setting the sslusessl MeterMaid option to 1 directs MeterMaid to expect that incoming connections will be SSL-enabled. Enabling this option requires that you also set the corresponding MeterMaid client sslusessl option.

59.2 metermaid_client options

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 metermaid_client:

```
msconfig> set metermaid_client.connectfrequency 10
```

Under metermaid_client, two types of named groups are also supported, remote_server and remote_table.

See the MeterMaid topic for further discussion of general MeterMaid operation. And see metermaid options for discussion of MeterMaid server options set under metermaid.

59.2.1 debug Option Under metermaid_client

The debug MeterMaid Client option enables debug output from the MTA client into SMTP log files.

59.2.2 connectfrequency Option Under metermaid_client

The connectfrequency MeterMaid client option, metermaid_client.connectfrequency, specifies that the client should attempt a connection to the MeterMaid server every connectfrequency seconds.

59.2.3 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).

59.2.4 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.

59.2.5 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.

59.2.6 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.

59.2.7 sslusessl Option Under metermaid_client

Setting the MeterMaid client option sslusessl to 1 directs the metermaid_client to connect to the MeterMaid server using SSL. This option also sets the default for the remote_server's sslusessl option.

Note that if configuring a MeterMaid client to use SSL to connect to a MeterMaid server, that MeterMaid server should also be configured to support SSL use; on the host where that MeterMaid server runs, see its metermaid.sslusessl option.

59.2.8 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.

59.2.9 remote_server

A `remote_server` named group may be set under `metermaid_client`.

59.2.9.1 `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.

59.2.9.2 `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.

59.2.9.3 `server_port` Option Under remote_server

The `server_port` MeterMaid client `remote_server` option specifies the TCP port to which the `metermaid_client` should connect for this `remote_server`.

59.2.9.4 `sslusessl` Option Under remote_server

Setting `sslusessl` to `1` in a MeterMaid client named `remote_server` group directs the `metermaid_client` to connect to the specified MeterMaid `remote_server` using SSL. Note that the default value for this option is the same as the value specified for the global `metermaid_client.sslusessl` option. If that value is set to `1`, this option will default to `1`.

59.2.10 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.

59.2.10.1 `server_nickname` Option

The `server_nickname` `metermaid_client.remote_table` option specifies the name of the group of `remote_server` entries that define a remote MeterMaid server.

Chapter 60 Notification messages

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.

60.1 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 *CONFIGROOT*`/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 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:

1. 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.

2. 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` and `ldap_domain_attr_capture` MTA options) is set on a user or domain, 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 just 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.

60.1.1 Message Store notifications that a user himself is overquota

Notifications to a user that that user himself 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*` 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.

A Messaging Server administrator may also manually trigger generation of a different form of quota warning message via the `imquotacheck` utility.

60.2 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. If the MTA is expanding a mailing list (looking up the mailing list name to determine the members of the list) and discovers a clearly invalid member address (*i.e.*, a syntactically invalid address, or an address claiming to be in a local domain that does not in actuality correspond to any configured user/mailbox), then the MTA will immediately generate a non-delivery report regarding that address back to the list owner (list report address). 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 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).

60.3 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/LotX5gy1oVEHL7SQ)"; report-type=delivery-status
Original-Recipient: rfc822;user1@domain.com


--Boundary_(ID_7qPwt/LotX5gy1oVEHL7SQ)
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)
                                                                   (13)(14)
--Boundary_(ID_7qPwt/LotX5gy1oVEHL7SQ)
```

```
Content-type: message/delivery-status                                    (15)

Original-envelope-id: 01MZZ7DIEUG400LXL0@host1.domain.com
Reporting-MTA: dns;host1.domain.com (tcp-daemon)                         (16)
                                                                         (17)
Original-recipient: rfc822;bogus@remote.com                              (18)
Final-recipient: rfc822;bogus@remote.com                                 (19)
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>                                          (20)
Received: from [129.158.87.66] by host1.domain.com (Sun Java System Messaging  (21)
 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 (17). 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. If the return_delivery_history MTA option is enabled (the default) and the original message has a history of delivery attempts, then some delivery history, up to the

limit specified by `history_to_return`, will be included here (after the `%R` recipient information), having the format:

```
Delivery attempt history for your mail:

time-stampdelivery-detailtime-stampdelivery-detail

...
```

14. Note that the `return_delayed.txt` file used when generating a delayed delivery warning often includes additional text *after* the `%R` recipient substitution, (and hence *after* any delivery history detail, as discussed in (13)), detailing how much longer the MTA will continue to attempt delivery. For instance, the distributed English language version of `return_delayed.txt` causes inclusion of text:

```
The mail system will continue to try to delivery your message
for an additional N time-units .
```

where `N` is derived from the `notices` final value, and `time-units` is derived from the `return_units` setting.

15. 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.

16. 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!

17. Some additional fields can potentially appear here, including X400-Content-identifier:, Content-identifier:, and UA-content-id:. As of MS 7.0, Arrival-date: and Future-release-request: also may potentially appear.

18. A set of information fields is output for each of the recipients of the original message, describing what happened for that recipient.

19. 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`.

20. The Return-path: records the envelope From of the original message.

21. 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.

60.3.1 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_*.*` 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_*.*` 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.

60.3.1.1 Customizing DSNs via the return_*.* 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_*.* files. The MTA will then use that set of return_*.* 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, %%.

60.3.1.1.1 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_*.* 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 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. (Note that the notary_decode MTA option controls any decoding and/or character set conversion of any MIME encoded non-US-ASCII material in such substituted header lines.)

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
```

60.3.1.1.2 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 60.1 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 imsimta return or the imsimta qm utility's return 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 notices 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 `notices` 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_TABLE:locale/C`. The MTA is also distributed with sets of `return_*.txt` files in other, language-specific directories under `IMTA_ROOT:lib/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.)

60.3.1.1.3 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 parameter 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).

60.3.1.1.3.1 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.

60.3.1.1.3.2 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 :
```

60.3.1.1.3.3 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 :
```

60.3.1.1.3.4 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.

60.3.1.1.3.5 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.

60.3.1.1.3.6 REASON (string)

The English language default is:

```
REASON=  Reason:
```

or for example a French language setting could be:

```
REASON=  Raison :
```

60.3.1.1.3.7 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 :
```

60.3.1.1.3.8 REMOTE_SYSTEM (string)

The English language default is:

```
REMOTE_SYSTEM=  Remote system:
```

or for example a French language setting could be:

```
REMOTE_SYSTEM=  Système distant :
```

60.3.1.1.3.9 *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.

60.3.1.1.3.10 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 60.2 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 `backoff` keyword value minus the %C time-queued-so-far.
%H			Substitute those header lines specified in the `return_header.opt` 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 `notices` 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 `backoff` 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 `notices` 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 `return_*.txt` file. That is, disable all % substitution sequence interpretation in the remainder of this template file.
%R	*see below*	`Recipient address:` *`address`* `Original address:` *`orcpt-value`* `Reason:` *`reason`* `Diagnostic code:` *`SMTP-error`* `Remote system:` *`name-and-details`*	The %R substitution causes output of a whole set of recipient-specific information, for each relevant envelope recipient, as a set of lines in *customizable-field-name field-value* 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 *customizable-field-name* text may be customized using a number of `return_option.opt` 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`	*MTA-error-text-for-x.y.z-status*	Substitute text explaining the "meaning" of the extended SMTP status *x.y.z*---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	

60.3.2 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 langauge-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:

`$I$T`*`langdir`*`,`*`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.

60.3.2.1 Customizing MDNs via the `disposition_*.*` 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_*.*` files, and optionally some override files and values. The MTA will then use that set of `disposition_*.*` 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, %%. For the %H substitution typically used in the `disposition_prefix.txt` file (to insert header lines from the original message into the generated MDN), note that any decoding and/or character set conversion of any MIME-encoded, non-US-ASCII material in those original header lines is controlled by the `notary_decode` MTA option.

60.3.2.1.1 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 60.3 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`	"`vacation :echo`" messages (or `mailAutoReplyMode: echo` 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 languge `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_TABLE:locale/C/`. The MTA is also distributed with sets of `disposition_*.txt` files in other, language-specific directories under `IMTA_ROOT:lib/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.)

60.3.2.1.2 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:

60.3.2.1.2.1 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.

60.3.2.1.2.2 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.

60.3.2.1.2.3 TEXT_CHARSET

When generating a vacation MDN in `:reply` format, and not using `:mime` (so not supplying all MIME header lines and any necessary MIME encoding), the vacation reason text may need to have charset conversion performed. The TEXT_CHARSET option specifies the output charset.

60.3.3 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 = *|*|*|*@*|* $C$|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|$2|$]ldap:///$2,o=internet?inetDomainBaseDN?sub?(objectClass=inetDomain)[
role.mapping:LDAP_USERS_LANGUAGE.rule = |DC|*@*|* $C|BDN|$0@$1|$2|$]ldap:///$2,o=internet?aliasedObjectName?sub?(objectClass=inetDomainAlias)[
role.mapping:LDAP_USERS_LANGUAGE.rule =
  |BDN|*|*|* $C|LANG|$]ldap:///$2?preferredLanguage?sub?(|(mail=$=$0$_)(mailAlternateAddress=$=$0$_)(mailEquivalentAddress=$=$0$_))[
role.mapping:LDAP_USERS_LANGUAGE.rule =
  |BDN|*|*|* $C|LANG|$]ldap:///$1,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|/$$I$Y$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,_base_dn_{
role.mapping:LDAP_USERS2_LANGUAGE.rule =
  |BDN|*|* $C|LANG|$]ldap:///$1?preferredLanguage?sub?(|(mail=$=$0$_)(mailAlternateAddress=$=$0$_)(mailEquivalentAddress=$=$0$_))[
role.mapping:LDAP_USERS2_LANGUAGE.rule = |BDN|*@*|* $C|LANG|$}$1,preferredLanguage{
role.mapping:LDAP_USERS2_LANGUAGE.rule =
  |LANG|* $CIMTA_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.

60.4 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 local Sieve filter owner letting them know that their Sieve filter has a syntax error.

The process channel, having had a notification message enqueued to it, will then run to 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-6.3P1 $~ 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 maytlsserver maysaslserver \
 saslswitchchannel tcp_auth missingrecipientpolicy 0 loopcheck
tcp-dsn-in-daemon

tcp_lmtpcs_dsn_out defragment lmtp multigate connectcanonical \
 fileinto @$4O:$U+$S@$D port 225 nomx single_sys subdirs 20 maxjobs 7 \
 pool SMTP_POOL dequeue_removeroute
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.

60.5 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` in turn out a special outbound channel, 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.

60.6 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*. (See also the new-in-MS-8.0 "`P`" records.) 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.

60.7 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 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.

60.8 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). In particular, in a modern Unified Configuration:

```
msconfig> show alias:root*
role.alias:root@&/IMTA_DEFAULTDOMAIN/.alias_entry = postmast
role.alias:root@&/IMTA_HOST/.alias_entry = postmast
msconfig> show alias:postmast*
role.alias:postmaster@&/IMTA_HOST/.alias_entry = postmast
```

which presumes that a valid user account (often the Messaging Server `admin` user) is provisioned with `postmast@/IMTA_DEFAULTDOMAIN/` as a `mailAlternateAddress` value.

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 61 Message tracking and recall

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.

61.1 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.

61.1.1 Memcache server setup

Setting up memcached or compatible server is beyond the scope of this document. That said, note that mecmached 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
```

61.1.2 MTA channel setup for message tracking and recall

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 messages relayed to other MTAs inside the deployment. (At a minimum, this tends to mean using a distinct `tcp_local` *vs.* `tcp_intranet` channel.) 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.)

61.1.3 MTQP server setup

An extended version of MTQP as 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. In legacy configuration:

```
mtqp smtp
mtqp-daemon
```

or in Unified Configuration:

```
msconfig> set channel:mtqp.officical_host_name mtqp-daemon
msconfig# set channel:mtqp.smtp
```

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:

- `maytlsserver` or `musttlsserver` 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 to 1, 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. In legacy configuration:

```
[SERVICE=MTQP]
PORT=1038
IMAGE=IMTA_BIN:mtqp
LOGFILE=IMTA_LOG:mtqp_server.log
STACKSIZE=2048000
```

or in Unified Configuration:

```
msconfig> set service:MTQP.image IMTA_BIN:mtqp
msconfig# set service:MTQP.tcp_ports 1038
msconfig# set service:MTQP.logfilename IMTA_LOG:mtqp_server.log
msconfig# set service:MTQP.stacksize 2048000
```

Chapter 62 TCP/IP channels

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 TCP/IP-channel-specific options. In legacy configuration, channel options were termed "channel keywords" and were set on channels in the `imta.cnf` file, whereas TCP/IP-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 esssential 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.

Note [1] 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.

62.1 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.maytlsserver (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.maytlsserver (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.maytlsserver (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.maytlsserver (novalue)
role.channel:tcp_tas.mustsaslserver (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 identnonenumeric inner loopcheck \
 maysaslserver maytlsserver nomx pool SMTP_POOL remotehost \
 saslswitchchannel tcp_auth smtp sourcespamfilter1 switchchannel
tcp-daemon

tcp_intranet allowswitchchannel dequeueremoveroute loopcheck maysaslserver \
 maytlsserver ms pool SMTP_POOL saslswitchchannel tcp_auth single_sys smtp \
 sourcespamfilter1
tcp_intranet-daemon

tcp_submit maytlsserver mustsaslserver saslswitchchannel tcp_submit smtp \
 sourcespamfilter1
tcp_submit-daemon

tcp_auth mustsaslserver smtp sourcespamfilter1
tcp_auth-daemon
```

```
tcp_tas allowswitchchannel deliveryfalgs 2 maytlsserver mustsaslserver 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.backoff = PT5M PT10M PT30M PT1H PT2H PT4H
role.channel:tcp_lmtpcs.connectcanonical (novalue)
role.channel:tcp_lmtpcs.defragment (novalue)
role.channel:tcp_lmtpcs.dequeueremoveroute (novalue)
role.channel:tcp_lmtpcs.fileinto = @$4O:$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.official_host_name = lmtpcs-daemon
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 @$4O:$U+$S@$D \
  port 225 nomx single_sys pool SMTP_POOL dequeue_removeroute \
  backoff "PT5M" "PT10M" "PT30M" "PT1H" "PT2H" "PT4H"
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.

62.2 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 `mustsaslserver` and `maytlsserver`, 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 `musttlsserver`.)

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.

62.2.1 BURL_ACCESS mapping table

As of Messaging Server 7.0, 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 62.1 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.
$;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 (and initial configuration will generate):

```
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*:*            $:A$:T$M$Y
! Allow BURL commands, connecting to (hosted domain) user's own mailHost
   *|*@*|imap://*;URLAUTH=submit+$1*%40$2*:*      $:A$: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*:*          $C$:A$|X-IMAP-HOSTS;$4|$Y$E
! Ditto for users in hosted domains
  *|*@*|imap://*@*/*;URLAUTH=submit+$1*%40$2*:*    $C$:A$|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.

62.2.2 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 submittor 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.
```

62.3 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 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.)

The MTA's LMTP server operates as a sort of combination of the MTA's SMTP server, and the `ims-ms` channel (the delivery into the Message Store portion of the `ims-ms` channel). In particular, the IMAP error statuses that can be encountered by the `ims-ms` channel, discussed further in ims-ms channel error messages, also are relevant for the LMTP server (and indeed, since the LMTP server in turn passes such errors back in the form of SMTP error responses to the LMTP client, they are reported to the LMTP client also).

62.3.1 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 `lmtp` 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.)

62.3.2 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 `LMTPSS` 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.identnonenumeric (novalue)
role.channel:tcp_lmtpss.lmtp (novalue)
msconfig> show dispatcher.service:*MTP*
role.dispatcher.service:LMTPSS.image = IMTA_BIN:tcp_lmtp_server
role.dispatcher.service:LMTPSS.logfilename = IMTA_LOG:tcp_lmtpss_server.log
role.dispatcher.service:LMTPSS.parameter = CHANNEL=tcp_lmtpss
role.dispatcher.service:LMTPSS.stacksize = 2048000
role.dispatcher.service:LMTPSS.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.l $Y
role.mapping:INTERNAL_IP.rule * $N
```

A recipe that creates this configuration would be:

```
description("Configure backend store accessible via LMTP");
keywords(["backend", "store", "LMTP"]);

# Back end's IP address
# Script will prompt for address if left black
#myIP = "10.133.152.193";
myIP = "";

# List of frontend machines that access this store via LMTP
# Script will prompt for addresses if left blank
#feIPs = ["10.133.152.192", "10.133.152.193"];
feIPs = [];

# Prompt for myIP and feIP if needed
if (length(myIP) <= 0) {
  myIP = read("Enter the IP address of this host: ");
}

if (length(feIPs) <= 0) {
  loop {
    ip = read("Enter the IP address of a frontend machine (<RET> if no more): ");
    exitif (ip == "");
    push(feIPs, ip);
  }
```

```
}

# Configure LMTP back end
if exists_channel("tcp_lmtpss") {
  warn("WARNING: tcp_lmtpss channel already exists, will not be added");
} else {
   print("INFO: Adding tcp_lmtpss channel\n");
   add_channel("tcp_lmtpss",
               ["flagtransfer", "",
                "identnonenumeric", "",
                "lmtp", "",
                "official_host_name", "tcp_lmtpss-daemon"]);
}

if exists_group("dispatcher.service:LMTPSS") {
  warn("WARNING: Dispatcher.service:LMTPSS group already exists, will not be created");
} else {
  print("INFO: Creating dispatcher.service:LMTPSS group\n");
  add_group("dispatcher.service:LMTPSS",
            ["image", "IMTA_BIN:tcp_lmtp_server",
             "logfilename", "IMTA_LOG:tcp_lmtpss_server.log",
             "parameter", "CHANNEL=tcp_lmtpss",
             "tcp_ports", "225",
             "stacksize", "2048000"]);
}

# Replace PORT_ACCESS mapping
print("INFO: Replace PORT_ACCESS mapping\n");
replace_mapping("PORT_ACCESS",
                ["TCP|*|225|*|*", "$C$|INTERNAL_IP;$1|$Y$E",
                 "TCP|*", "$N500 Do not connect to this machine"]);

# add PORT_ACCESS mapping entries
internal_ip = ["${::1}", "$Y",
               "127.0.0.1", "$Y",
               "*", "$N"];

# list of IP addresses
ipaddrs = [feIPs];
push(ipaddrs, myIP);

loop {
  exitif (ipaddrs == []);
  ip = pop(ipaddrs);
  push(internal_ip, "$Y");
  push(internal_ip, ip);
}

print("INFO: Replace INTERNAL_IP mapping\n");
replace_mapping("INTERNAL_IP", internal_ip);
```

In legacy configuration, this would be defined as a `tcp_lmtpss` channel in the `imta.cnf` file:

```
tcp_lmtpss flagtransfer identnonumeric lmtp
tcp_lmtpss-daemon
```

and a Dispatcher definition of the LMTPSS server in the `dispatcher.cnf` file:

```
!
! rfc 2033 LMTP server - store
!
[SERVICE=LMTPSS]
PORT=225
IMAGE=IMTA_BIN:tcp_lmtp_server
LOGFILE=IMTA_LOG:tcp_lmtpss_server.log
PARAMETER=CHANNEL=tcp_lmtpss
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
```

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
```

Note that this configuration is only appropriate for back ends where delivery is only performed by LMTP. In particular, the `PORT_ACCESS` and `INTERNAL_IP` mapping mapping tables shown here would not be appropriate for back ends that also accept mail via SMTP.

62.3.2.1 LMTP server options

The `loglevel` option may be set under the `tcp_lmtp_server` group to override for LMTP server purposes any MTA-wide setting (`mta.logfile.loglevel`) of the nslog log level (controlling any Message Store logging triggered from the LMTP 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.

62.4 TCP/IP-channel-specific options

TCP/IP channels support, in addition to the usual channel options, a number of TCP/IP-channel-specific options.[1] 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 `IMTA_TABLE: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 `IMTA_TABLE: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 TCP/IP-channel-specific 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 its options (in legacy configuration, its option file) only when it is first started, hence will not see changes while it continues to run, and its running time may be on 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).

[1] 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.

62.4.1 TCP/IP-channel-specific options: PERMANENT_ERROR_STRING_552 (string)

Important note: Prior to MS 8.0.1.3, the name of this option was 552_PERMANENT_ERROR_STRING. Unfortunately the leading digit "5" in the option makes this option unworkable in unified configuration due to XML token name restrictions, so the name of the option had to be changed to PERMANENT_ERROR_STRING_552. Both names work in 8.0.1.3 and later legacy configurations.

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 the last line of that response (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 error 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:

```
PERMANENT_ERROR_STRING_552=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.

As of 8.0.1.3 multiple patterns can also be specified, separated by vertical bars ("|").

62.4.2 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.

62.4.3 TCP/IP-channel-specific options: ALLOW_RECIPIENTS_PER_TRANSACTION (integer)

This SMTP/LMTP server option set s 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).

62.4.4 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
```

62.4.5 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.

62.4.6 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.

62.4.7 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).

62.4.8 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.

62.4.9 TCP/IP-channel-specific options: AUTH_DEBUG (string)

(New in Messaging Server 7.4-18.01.) The AUTH_DEBUG 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 AUTH_DEBUG is set, it overrides the debugkeys Base option 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.

62.4.10 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.

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 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.)

See the Base certmap options for general configuration of certificate mapping, as needed for EXTERNAL authentication via client certificates.

62.4.11 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.

62.4.12 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).

62.4.13 TCP/IP-channel-specific options: BANNER_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies how long to wait to receive the SMTP/LMTP initial banner from the server. The default value is 2 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.14 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.

62.4.15 TCP/IP-channel-specific options: BANNER_PURGE_DELAY (integer)

(New in 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.

62.4.16 TCP/IP-channel-specific options: BUFFER_SIZE (integer)

(New in 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).

Messaging exceeding the limit will be written to a temporary file and the temporary file will then be mapped into memory.

In unified configuration the option can be set as follows:

```
msconfig> set channel:tcp_lmtpss.options.buffer_size 1000000
```

62.4.17 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 identnonenumeric 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.

62.4.18 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.

62.4.19 TCP/IP-channel-specific options: CLIENT_STACK_SIZE (integer)

New in MS 8.0.1.3. This SMTP/LMTP client option takes an integer number indicating the stack size to allocate for SMTP client threads. The default value is 1024*1024 = 1048576. It is strongly recommended that this value only be increased.

62.4.20 TCP/IP-channel-specific options: COMMAND_RECEIVE_TIME (integer)

This SMTP/LMTP server option specifies 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 minutes

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.21 TCP/IP-channel-specific options: COMMAND_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies 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 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.22 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 option; 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.

62.4.23 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 "Oracle Communications Messaging Server <version-number> (<build-date>)" is used. This option is thus a complement to the (also not recommended) received_version MTA option.

62.4.24 TCP/IP-channel-specific options: DATA_RECEIVE_TIME (integer)

This SMTP/LMTP server option specifies how long to wait to receive each line of data during the DATA phase of an SMTP/LMTP dialogue. The default is 5 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.25 TCP/IP-channel-specific options: DATA_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies how long to spend transmitting data during an SMTP/LMTP dialogue. The default is 10 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.26 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 consistute a breach of security for some sites. Settting 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.

62.4.27 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 consistute 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.

62.4.28 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 constitue 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).

62.4.29 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.

62.4.30 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`.

62.4.31 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. Settting 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.

62.4.32 TCP/IP-channel-specific options: DOT_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies how long to spend transmitting the dot (period) terminating the data in an SMTP/LMTP dialogue. The default is 10 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.33 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.

62.4.34 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 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.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.35 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.

62.4.36 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.

62.4.37 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.

62.4.38 TCP/IP-channel-specific options: KILLED_IP_TIMEOUT (integer seconds)

The SMTP server option KILLED_IP_TIMEOUT may be used to specify an the timeout value, in seconds, for entries made in the internal killed IP hash table associated with the imsimta connkill -ip command. The default is twice the value of the COMMAND_RECEIVE_TIME TCP/IP channel-specific option.

Note that the timeout value affects storage of all IP entries, however, the setting is especially relevant when the -sticky switch is used.

62.4.39 TCP/IP-channel-specific options: KILLED_USER_TIMEOUT (integer seconds)

The SMTP server option KILLED_USER_TIMEOUT may be used to specify an the timeout value, in seconds, for entries made in the internal killed user hash table associated with the imsimta connkill -user command. The default is 600 seconds.

Note that the timeout value affects storage of all user entries, however, the setting is especially relevant when the -sticky switch is used.

62.4.40 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.

62.4.41 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 62.2 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.

62.4.42 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.

62.4.43 TCP/IP-channel-specific options: MAIL_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies how long to spend transmitting the SMTP command MAIL FROM:. The default is 10 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.44 TCP/IP-channel-specific options: MAILBOX_BUSY_FAST_RETRY (integer)

(New in 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 + rand() % (120 * n) seconds. In release 7.0 through 7.0.5.31.0, the formula used is n + rand() % (4 * n) seconds.)

As of MS 8.0.2.2, this option is also supported by ims-ms channels.

62.4.45 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.

62.4.46 TCP/IP-channel-specific options: MAX_B_ENTRIES (integer)

New in 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 MS 6.2) `disconnectbadcommandlimit` channel option.

62.4.47 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.

62.4.48 TCP/IP-channel-specific options: MAX_H_ENTRIES (integer)

New in MS 8.1. This TCP/IP channel SMTP server option specifies the maximum number of "J" `mail.log*` entries to write during a single SMTP connection session. 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 VRFY and EXPN commands, so that deliberate abuse of these commands 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 H record for this session)".

62.4.49 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.

62.4.50 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 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 MS 6.2, the "Rejected" counter will continue to be incremented for each additional rejected attempt during the session.)

62.4.51 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.)

62.4.52 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 is 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.

62.4.53 TCP/IP-channel-specific options: OPEN_CONNECTION_TIME (integer)

This option controls how long the SMTP/LMTP client waits for a TCP/IP connection to open. The default is 2 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.54 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.

62.4.55 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 `smtpproxypassword` 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.
```

62.4.55.1 XPEHLO proxy validation: `proxy_hash_algorithm` (hash algorithm name)

New in MS 8.1.0.3. The `proxy_hash_algorithm` MTA option controls what hash algorithm the MTA uses to authenticate the contents of an XPEHLO command. The value should be a hash algorithm supported by the MTA, one of MD2, MD4, MD5, SHA1, SHA256, SHA512, MD128, or MD160. MD5 is the default. Note that the setting of this option must be the same across a deployment for successful operation of the XPEHLO command.

62.4.56 TCP/IP-channel-specific options: RCPT_TRANSMIT_TIME (integer)

This SMTP/LMTP client option specifies how long to spend transmitting the SMTP command RCPT TO:. The default is 10 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.57 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.

62.4.58 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 (MS 6.3 and earlier) or in JS Messaging Server 7.0, `$DATAROOT/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 noticable 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.

62.4.59 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 20 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
```

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.60 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 threshhold levels are exceeded. That is, the SMTP server can in gradated 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 threshhold 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.

62.4.61 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
```

62.4.62 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.

62.4.63 TCP/IP-channel-specific options: STATUS_DATA_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies 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 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

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.

62.4.64 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.

The base unit time is in units of minutes by default; as of MS 8.0.2.3 this may be adjusted with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option.

62.4.65 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.)

The base unit time is in units of minutes by default; as of MS 8.0.2.3 this may be adjusted with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option.

62.4.66 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.)

The base unit time is in units of minutes by default; as of MS 8.0.2.3 this may be adjusted with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option.

62.4.67 TCP/IP-channel-specific options: STATUS_MAIL_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies 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 minutes

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.68 TCP/IP-channel-specific options: STATUS_RCPT_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies how long to wait to receive the SMTP response to a sent RCPT TO: command. The default value is 10 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the `TIMEOUT_MULTIPLIER` TCP/IP-channel-specific option. In particular, setting `TIMEOUT_MULTIPLIER` to 1 will change the units of this option to seconds.

62.4.69 TCP/IP-channel-specific options: STATUS_RECEIVE_TIME (integer)

This SMTP/LMTP client option specifies 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 minutes.

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.70 TCP/IP-channel-specific options: STATUS_TRANSMIT_TIME (integer)

This SMTP/LMTP server option specifies how long to spend transmitting the SMTP/LMTP reply to an SMTP/LMTP command. The default value is 10 minutes

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.71 TCP/IP-channel-specific options: TLS_NEGOTIATION_TIME (integer)

This SMTP/LMTP client option specifies how long the SMTP/LMTP client and server will wait for the opposite end during TLS negotiation. The default value is 1 minute.

The default units for this option are minutes. The units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.72 TCP/IP-channel-specific options: TIMEOUT_MULTIPLIER (integer)

This SMTP/LMTP client/server TCP/IP-channel-specific option specifies a multiplier value that is applied to various other TCP/IP-channel-specific timeout options. While any positive value is allowed, settings should be limited to 60 (the default), which causes the various associated timeout options to be interpreted in minutes, or 1, which causes the timeouts to be interpreted in seconds.

62.4.73 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.

See also the $G flag of the PORT_ACCESS mapping table, which can be used to selectively enable TRACE_LEVEL=2 only for particular incoming SMTP/SMTP SUBMIT/LMTP connections.

62.4.74 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
```

62.4.75 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 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
```

The default units for this option are minutes. As of MS 8.0.2.3, the units of this option may be changed with the TIMEOUT_MULTIPLIER TCP/IP-channel-specific option. In particular, setting TIMEOUT_MULTIPLIER to 1 will change the units of this option to seconds.

62.4.76 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).

62.5 DEQUEUE_ACCESS mapping table

New in 8.0.2.3 is the DEQUEUE_ACCESS mapping table, which is consulted at the beginning of message dequeue processing.

The default probe for the DEQUEUE_ACCESS mapping is of the form:

channel|*filename*|*queue-time*|*envelope-from*|*auth-parameter*|*auth-sender*|*priority*|*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, *auth-sender* is the mail address associated with any authentication identity used to submit the message, *priority* is the MT-priority value for the message, and *domain* is the DNS name of the server to which the message is to be sent. Note that the mapping_paranoia MTA option, if set, will cause any vertical bar characters that would have been in the *envelope-from*, *auth-parameter*, *auth-sender*, or *domain* fields to be replaced by the specified character.

As of MS 8.0.2.3, the $R input flag will be set if the message is being accessed in read-only mode by a utility such as imsimta qm.

If bit 11, value 4096, of the include_conversiontag MTA option is set the probe changes to include the conversion tag for the message:

channel|*filename*|*queue-time*|*envelope-from*|*auth-parameter*|*auth-sender*|*priority*|*conversio*

If bit 1, value 2, of the include_retries MTA option is set the probe changes to include the count of previous retries for the message:

channel|*filename*|*retry-count*|*queue-time*|*envelope-from*|*auth-parameter*|*auth-sender*|*priori*

The DEQUEUE_ACCESS mapping should return a list of values delimited by vertical bars ("|").

(New in MS 8.1.0.6) The $> flag is used to indicate that a smartsend logging result has been returned. A value is consumed from the returned string and appended to the smartsend log value. Note that this value is only logged if the log_smartsend MTA option is set.

If $N is set by the mapping the current dequeue action is aborted without incrementing the retry count. If the $S flag is also set a new backoff time in seconds, expressed as an integer, is read from the mapping result string.

62.6 AUTH_ACCESS mapping table

New in 7.0.5 is the AUTH_ACCESS mapping table, which provides the means to exercise increased control over SMTP session characteristics. 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 default 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_paranoia 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.

New in MS 8.0.2.2, If bit 11, value 2048, of the include_conversiontag MTA option is set the probe changes to include the conversion tag for the message:

channel|*filename*|*queue-time*|*envelope-from*|*auth-parameter*|*username*|*conversiontag*|*domain*

New in MS 8.0.2.3, if bit 0, value 1, of the include_retries MTA option is set the probe changes to include the count of previous retries for the message:

channel|*filename*|*retry-count*|*queue-time*|*envelope-from*|*auth-parameter*|*username*|*domain*

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 output flags.

Table 62.3 AUTH_ACCESS mapping output flags

Flag	Fields	Description
$U		Enable SMTP client debugging for this transaction. No fields are consumed.
$n	*error-text*	(New in MS 8.0.2.3.) Abort this connection attempt for this message. One field is used to specify the error text. No further processing of the mapping result is performed.
$N	*error-text*	Permanently fail the first recipient of the message and return a DSN. The delivery operation continues with the next recipient. The entire message fails if there is only one remaining recipient. One field is used to specify the error text. No further processing of the mapping result is performed.
$F	*env-from*	Override the message's envelope From address. One field is used to specify that address.
$A	*auth-param*	Override the MAIL FROM AUTH= parameter. One field is used to specify the override value.
$Q	*authorization*|*username*|*password*	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, *vs.* causing an SMTP permanent failure (bounce) when mustsaslclient is in effect.)
$D	*host*	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 $S to enable smtps: use.)
$P	*port*	Override the value of the port 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 $S to enable smtps: use.)
$S		Enables the use of TLS (smtps:). No fields are consumed. Note that this flag enables smtps: use even when $D or $P has also been used.
$X		Disable MX lookups for connection establishment (overriding any channel *mx channel option setting). No fields are consumed.
$o		(New in MS 8.1.0.2) Treat nonauthoritative DNS lookup failures as temporary failures.
$O		(New in MS 8.1.0.2) Treat all DNS lookup failures as temporary failures.
$M		Enable MX lookups for connection establishment (overriding any channel *mx keyword setting); the effect is randommx MX record use. No fields are consumed.
$B	*lastresort-server*	Set a lastresort value, overriding the value specified by the lastresort channel option. One field is consumed as the lastresort server name.
$!	*min-version*|*max-version*	New in MS 8.1.0.1. Set tlsminversion and tlsmaxversion value, overriding the values specified by the corresponding channel options. Two field are consumed as the minimum and maximum values. Either field may be left blank to avoid affecting the corresponding value.
$T		Force musttlsclient on for this session. No fields are consumed.
$H		Disable TLS for this session. No fields are consumed.
$G		Force mustsaslclient on for this session. No fields are consumed.
$I		Disable SASL for this session. No fields are consumed.
$Z		(New in MS 8.1.0.1) Disable CHUNKING for this session. No fields are consumed.
$J	interface-address	(New in 8.0.2.2) Use the specified value as the source IP address. A single field containing the IP address to use is consumed. If a connection is currently open it will be closed unless the interface address has not changed.
$+.	host-name	(New in 8.0.2.3) Use the specified value as the host name in any EHLO/HELO/LHLO command that the client issues. A single field containing the banner host name is consumed.
$V	skip-count	(New in 8.0.2.3) Specify a new skip count to be encoded in the queue file name for the message. This flag is used by the smartsend plugin and is not

		intended for other purposes. One field containing an unsigned integer skip value is consumed.
$+R	attempt-count	(New in 8.0.2.3) Specify an override value for the ATTEMPT_TRANSACTION_PER_SESSION TCP/IP-specific channel option. A single field containing the override value is consumed.
$(	max-mx-count	(New in 8.0.2.3) Specify an override value for the MAX_MX_RECORDS TCP/IP-specific channel option. A single field containing the override value is consumed.
$+%	backoff-time	(New in 8.0.2.3) Specify an override value for the backoff time in the event this delivery attempt fails with a temporary error. A single field containing the time in seconds is consumed.
$*	header1,header2,...	(New in 8.1.0.1) Specifies a list of header fields to log in any transaction log entries that are generated. A single field is consumed.
$>	smartsend-log-string	(New in 8.1.0.6) Specified a string to append to the smartsend log value. A single field is consumed.
$,	deaccess-parameter-string	(New in 8.0.2.3) Specify the deaccess parameter string to pass to the AUTH_DEACCESSS mapping. Note that this consumes all remaining arguments in the mapping result stirng.
$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		Description
$:|		Match only if external material (*e.g.*, 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
$:S		(New in MS 8.0.2.3) Match only if a connection to the destination for this message is already open and is going to be reused.
$;S		(New in MS 8.0.2.3) Match only if no SMTP connection is open or if one is open it is going to be closed prior to processing this message.
$:T		(New in MS 8.0.2.3) Match only if STARTTLS is allowed or required for this connection
$;T		(New in MS 8.0.2.3) Match only if STARTTLS is not allowed for this connection.
$:U		(New in MS 8.0.2.3) Match only if smtps: is allowed or required for this connection
$;U		(New in MS 8.0.2.3) Match only if smtps: is not allowed for this connection.

As of MS 8.1.0.1, there are also a number of input flags:

Table 62.4 AUTH_ACCESS mapping input flags

Flag	Description
$S	Set if a connection is currently open that has the potential to be reused.
$D	Set if the destination host associated with the currently open connection matches the destination host for the current message. $D is only set if $S is also set.
$T	Set if current settings allow or require TLS.
$U	Set if current settings allow or require smtps.

The AUTH_ACCESS mapping table can be used for a variety of purposes, including various special-purpose, targetted, 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:

```
AUTH_ACCESS

 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 AUTH_ACCESS mapping example: third party submission, 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 AUTH_ACCESS mapping example: third party submission, 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 AUTH_ACCESS mapping example: Excerptoflocal user entry in user/group LDAP and AUTH_ACCESS mapping example: Excerpt of remote identity entries in alternate (external) LDAP 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.

62.6.1 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@remote1.domain.com
mailRemoteIdentity: adam.brown@remote2.domain.com
```

62.6.2 AUTH_ACCESS mapping example: Excerpt of remote identity entries in alternate (external) LDAP

```
Under the remote1.domain.com portion of the "external" LDAP DIT:

mail: abrown@remote1.domain.com
username: remote1-username
password: remote1-password
submittor: 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
submittor: adam.brown@local.domain.com
```

Another important aspect to consider is 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. AUTH_ACCESS mapping example: third party submission 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.

62.6.3 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
! "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-baseDN>|<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-baseDN>|<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"'
```

62.7 AUTH_DEACCESS mapping table

New in MS 8.0.2.3. The AUTH_DEACCESS mapping forms a pair with the AUTH_ACCESS mapping table. It is only called when the AUTH_ACCESS mapping has been called and has set a deaccess parameter string.

The call occurs immediately after the final connection associated with the AUTH_DEACCESS mapping has been closed. Note that this may occur right after the message associated with the AUTH_ACCESS call has been processed or it may occur many messages later as a result of connection reuse. And since AUTH_ACCESS is called for every message, there may have been

multiple intervening calls to the AUTH_ACCESS mapping, including a final one associated with setting up a new connection.

This mapping is intended to be used to release resources allocated by the AUTH_ACCESS mapping, typically through the use of a mapping callout. More specifically, AUTH_ACCESS can now be used to allocate some connection-related resource, which can then be used by one or more connections used to deliver the current message and possibly subsequent messages. The AUTH_DEACCESSS mapping is called when the last connection associated with this activity is finally closed.

The probe for the AUTH_DEACCESS mapping is of the form:

channel|*filename*|*deaccess-parameter-string*

Note that the *deaccess-parameter-string* may consist of multiple |-separated sections.

At present the result of the AUTH_DEACCESS is ignored.

62.8 MX_ACCESS mapping table

New in MS 8.1.0.1, the MX_ACCESS mapping table provides the means to control and/or override MX lookup operations.

The MX_ACCESS access mapping table is consulted during SMTP/LMTP client operations just prior to performing the MX lookup on the deatination domain. It thus provides a means of changing the server hosts for a given domain. c

The MX_ACCESS mapping table probe has the following default format:

source-channel|*to-address*|*domain*

Or if bit 16 (value 65536) of the include_conversiontag MTA option is set, then the format is:

source-channel|*conversion-tag*|*to-address*|*domain*

Here *source-channel* is the channel from which the message is being dequeued, *conversion-tag* is a comma-separated list of the conversion tags associated with the message, *to-address* first recipient address, and *domain* is the domain the message is being sent to.

The mapping can set the flags shown in Table of MX_ACCESS mapping flags.

Table 62.5 MX_ACCESS mapping flags

Flag	Description
$U	Enable channel debugging.
$F	Force a "host not found" error, as if the MX lookup failed because the domain does not exist.
$N	Force a temporary DNS error, as if the MX lookup failed due to a transient problem.
$T	Change the destination host to the mapping result string.

$Y	Force the MX result to be the list of hosts specified in the mapping result string. The order of this list will be randomized if mxrandom is in effect.

Note that at present all of the mapping actions are mutually exclusive, that is, it only makes sense to specify one of $F, $N, $T, or $Y.

62.9 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 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 62.6 IP_ACCESS mapping flags

Flag	Description
$N	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.
```

62.10 SASL_ACCESS mapping table

The new-in-MS-8.0.2 SASL_ACCESS mapping probe has the form:

operation|*transport-info*|*app-info*|*src-chan*|*bad-auth-count*|*username*|*admin-type*|*statusc*

Here *operation* is currently always "AUTHENTICATE". 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. *src-chan* is the currently selected source channel, *bad-auth-count* is a count of the number of previous failed authentication attempts on this connection, and *username* is the user name that the authentication operation bound to, if any. The *admin-type* is the administrator type from the account the authentication operation bound to, which will be one of:

- N/A

- NONE
- FULL
- READONlY
- DOMAIN
- UNKNOWN

Finally, *statuscode* is the status returned by the authentication operation attempt. The possible values are listed in the MTAauth errors table.

The SASL_ACCESS mapping template can set any of the following flags:

Table 62.7 SASL_ACCESS mapping table flags

Flag	Meaning
$U	Enable debugging
$<*string*	Send string to syslog
$>*string*	Send string to syslog if access is explicitly rejected
$N*string*	Reject access with authentication result specified by *string*
$F*string*	Synonym for $N
$X*string*	Override authentication result string with specified *string*
$D*n*	Delay sending response by *n* centiseconds
$A*n*	Update session protocol delay to be *n* centiseconds
$*	Disconnect session after sending response

62.11 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|*app-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. 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, but note that the *app-info* will be limited in cases where TLS negotiation occurs before an EHLO/HELO command is issued.

Table 62.8 TLS_ACCESS mapping flags

Flag	Description

$<*string*	Send a *string* to syslog.
$>*string*	Send a *string* to syslog if a negotiation failure is forced ($F or ($N is also set).
$N	(New in 8.1.) Force TLS failure and close the connection, even though negotiation succeeded, with error string *string*. On outgoing connections $N differs from $F in that it allows fallback to an unencrypted connection.
$F*string*	Force TLS failure and close the connection, even though negotiation succeeded, with error string *string*.
$S*string*	(Incoming connections only.) Switch to channnel *string*.
$B	(New in 8.1.) (Incoming connections only.) Block any subsequent channel switch by the tlsswitchchannel channel option.

+ To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

62.12 SMTP_ACTIONS mapping table

New in MS 8.1, the SMTP_ACTIONS mapping table can be used to process SMTP failure responses and successful delivery of messages in the SMTP client and customize the MTA's handling in various ways.

If present, the SMTP_ACTIONS mapping is called whenever a failure is returned by the remote SMTP server. The mapping probe format is:

port_access-probe-info|*app-info*|*channel*|*domain*|*conversion-tags*|*what*|*reply*

New in MS 8.1.0.1, the *conversion-tags* are only included in the probe if bit 15, value 32768 of the include_conversiontag MTA is set.

Here *port_access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe (see PORT_ACCESS mapping table) and *app-info* is application-specific information, which here will usually be SMTP/*domain*/*mxhost* in the case of messages being relayed over SMTP and SMTP/*domain*/*mxhost*/TLS-*crypto-info*/*HELO-name* in the case of messages being relayed via SMTP over TLS. *channel* is the channel from which the message is being dequeued, *domain* is the destination domain, *mxhost* is the DNS name of the server to which the message is being relayed, *what* is the name of the SMTP command that produced the error (INITIAL, EHLO/HELO, STARTTLS, AUTH, MAIL, RCPT, DATASTART, DATAEND, RSET, QUIT), and *reply* is the SMTP reply string.

As previously noted, the SMTP_ACTIONS is only called when a command fails and unconditionally for the end of data (DATAEND) response.

Note also that the mapping_paranoia MTA option, if set, will cause any vertical bar characters that would have been in the *reply* field to be replaced by the specified character.

The following flags can be set by the mapping:

Table 62.9 SMTP_ACTIONS mapping flags

Flag	Fields	Description

$T		Treat the error as temporary even if a 5YZ code was returned.
$K		Treat the error as permanent even if a 4YZ code was returned.
$+#	*syslog-text*	Send the specified text to syslog. One return value is consumed.
$S	*comment-text*	Add the specified comment-text as a parenthetical comment to the logged diagnostic value. One return value is consumed.
$%	*backoff-value*	Replace the current backoff time with the specified value, expressed as an integer number of seconds. One return value is consumed.
$<		Minimize the supplied new backoff value with the current backoff value.
$>		Maximize the supplied new backoff value with the current backoff value.
$A	*header-text*	Add the specified header-text as a set of per-recipient DSN fields (see RFC 3464 section 2.3) to any DSN that is generated. One return value is consumed; is it processed as a tilde-separated list of header fields.

62.13 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.

62.13.1 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 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
```

62.14 Blocking SMTP relaying

One application of the ORIG_SEND_ACCESS mapping table, along with appropriate configuration of channels with switchchannel and saslswitchchannel 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 and initial configuration 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 saslswitchchannel 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.

62.14.1 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
```

62.15 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 `ETRN_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 `ETRN_ACCESS` mapping table. Probes 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.) If the mapping table returns a $N, $n, $F, or $f, then the ETRN 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 $D*channel-name* or $d*channel-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).

62.16 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 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
```

62.17 Authentication errors

Table 62.10 HULA error status

Name	Text	Meaning
HULA_FAIL	Internal authentication error	
HULA_NOMEM	Not enough memory available	
HULA_BUFOVER	Internal buffer overflow	
HULA_NOMECH	Mechanism not available	
HULA_BADPROT	Invalid authentication protocol	
HULA_NOTDONE	Authentication not complete	
HULA_BADPARAM	Invalid parameter supplied	
HULA_TRYAGAIN	Transient failure -- try authentication again	
HULA_BADMAC	Message corrupted in transit	
HULA_BADSERV	Server failed authentication	
HULA_WRONGMECH	Mechanism doesn't support requested feature	
HULA_BADAUTH	Authentication failed	
HULA_NOAUTHZ	Not authorized to login asspecified user	
HULA_TOOWEAK	A stonger authentication mechanism is required	
HULA_ENCRYPT	Encryption required	
HULA_TRANS	A transition is needed to use the specified mechanism	
HULA_EXPIRED	Password expired	
HULA_DISABLED	Account disabled	Roughly speaking, a domain status of "`disabled`" or user status of "`disabled`". (More precisely, a domain status or user status that is not specially handled in another way: so for instance unrecognized status values also result in this error.)
HULA_NOUSER	User not found	
HULA_PWLOCK	Password locked	
HULA_NOCHANGE	No change was made	
HULA_NOTINIT	SASL library not initialized	

HULA_UNAVAIL	Service temporarily unavailable	Can't connect, or have trouble communicating with, the authentication server (*e.g.*, LDAP server).
HULA_NOVERIFY	Missing server authentication entry for user	
HULA_WEAKPASS	Password fails site security policy, try another	
HULA_NOUSERPASS	User supplied password not permitted	
HULA_INPROGRESS	Operation in progress	
HULA_SVRACCESS	Access control filter on server forbids connection	
HULA_DOMACCESS	Access control filter on domain forbids connection	
HULA_USERACCESS	Access control filter on user forbids connection	

Chapter 63 BSMTP channels

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.*

63.1 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 bsmtp daemon host1
BSOUT-REMOTE1
```

```
bsout_remote2 smtp master user bsmtp daemon host2
BSOUT-REMOTE2...

bsout_remoteN smtp master user bsmtp 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 remote1, remote2, ... remoteN and REMOTE1, REMOTE2, ..., REMOTEN are arbitrary and need just be distinct from one another.

With the above definitions, the channel bsout_remote1 will bundle up its BSMTP parcels and send them on to the fixed address bsmtp@host1. Likewise for the remaining BSOUT channels.

The rewrite rules appear as:

```
domain1      $U%$H@BSOUT-REMOTE1$Nbsout_remote1
.domain1     $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1
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

  bsmtp@host0     bsmtp@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 bsmtp@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_remote1
.domain.co.uk    $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1

...

bsin_gateway smtp
bsin.hub.domain.com

bsout_remote1 smtp master user bsmtp 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_remote1
.domain.com   $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1

...

bsin_gateway smtp
bsin.athena.domain.co.uk

bsout_remote1 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_remote1` 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_remote1` tag in the domain.co.uk rewrite rules, those rewrite rules will be ignored when the `bsout_remote1` 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.

63.2 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.

Chapter 64 ims-ms channels

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.

64.1 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 inabilities 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 "l" channel's official host name, l.official_host_name in Unified Configuration, (or in legacy configuration, the first name on the second logical line of the c"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:l.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.

64.1.1 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.

64.2 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.*`config-root/ims-ms_option`.

64.2.1 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.)

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 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.

64.2.2 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.

64.2.3 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.

64.2.4 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 exitting. 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.

64.2.5 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.

64.3 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.

64.4 ims-ms channel debugging and error logging

Channel debugging output for the `ims-ms` channel goes to two (or three) places, the *`channel-name`*`_master.log-*` file, and the `imta` file, which is an NSLOG file, and as of MS 6.2, if message tracing is enabled (if the `messagetrace.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, including but not limited to the generation of nondelivery notifications, enabled for instance via the `master_debug` channel option, goes to a *`channel-name`*`_master.log-*` file. (Note that setting `master_debug` also forces the debug level controlled by `ims-ms.options.DEBUG` to at least 4, and as of MS 8.0.2, forces the `mta.logfile.loglevel` option setting to "debug".)

All 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.logfile.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 in the ims-ms channel option file in legacy configuration) to in fact get included in the imta log file, mta.logfile.loglevel (Unified Configuration) or logfile.imta.loglevel (legacy configuration) must hence be set to debug.

As of 8.0.1.2 the ims-ms.options.DEBUG option value defaults to 2. In earlier 8.0 versions the value is forced to 10 and the option setting is ignored. In 7.0.5 and earlier the default is 0, so either the channel-specific option or the master_debug channel option must be set in addition to mta.logfile.loglevel for debug output to appear.

A number of conditions can cause the ims-ms channel to fail to initialize. When this happens a critical error is sent to the imta file and the process exits with a nonzero status. Possible initialization failures include, but are not limited to:

PMDF_CHANNEL environment variable not defined	The PMDF_CHANNEL environment variable is used by the job controller to communicate the channel whose messages are to be processed. This error indicates that the variable is not set. This is usually an indication that the program has been run manually.
Failed to initialize the MTA	The MTA failed to initialized. This is usually the result of a configuration error of some sort
ims_master requires store.enable = 1	Self-explanatory.
ims_master requires store.dbtype = "bdb"	Self-explanatory.
inetu_uginit failed	The process was unable to initialize the LDAP client subsystem.
Failed to initialize the store API	The store failed to initialize. Additional errors may be logged providing more specific information

64.5 ims-ms channel error messages

MTA channels attempting to deliver to the Message Store, *i.e.*, ims-ms channels or tcp_lmtpcs* channels, can encounter several different sorts of issues: initialization errors (errors initializing the MTA, the Message Store, Message Store notify plugin code (if notifytarget named group options are set in Unified Configuration, or the local.store.notifyplugin configutil parameter is set in legacy configuration), domain map code, or LDAP pool code) many of which will cause the channel to abort, subsequent Message Store errors, subsequent trouble getting LDAP information about a domain or user, or IMAP error statuses when attempting to deliver particular messages. When encountering a problem doing an LDAP lookup for a domain or user while attempting to deliver a message, or an IMAP error status on a particular message, the MTA channel keeps running (and moves on to attempting to deliver any other messages awaiting delivery).

In particular, the IMAP error statuses reported by an `ims-ms` channel or LMTP server, when it attempts to access or manipulate the Message Store, are not specific to such channels, but rather are general IMAP error statuses (which may also be seen in other contexts, such as IMAP e-mail client attempts to access or manipulate the Message Store). Additional details on the underlying Message Store issue that gave rise to an IMAP error status may be available in the Message Store NSLOG logging for the component which encountered the issue; in the case of the `ims-ms` channel or LMTP server, see the `imta` NSLOG file discussed in ims-ms channel debugging and error logging.

Some IMAP error statuses are considered "permanent" errors: a message will be immediately bounced, if such an error is encountered. Other IMAP error statuses are considered "temporary" errors: a message will be retained in the MTA channel queue (`ims-ms` or LMTP client `tcp_lmtpcs*` channel) for further delivery attempts. See IMAP error statuses for a full listing of IMAP errors, or see IMAP error statuses relvant to the MTA for the subset of IMAP errors relevant to the MTA, and in particular relevant to the `ims-ms` and LMTP channels. (Note that in the case of LMTP, the IMAP error status is encountered by the LMTP server on a Message Store back end, which then reports the error to the LMTP client running on a "front end" MTA. So in the LMTP case, such errors may be reported in the LMTP server's message transaction log file, in an LMTP server log file, and also as a reported LMTP error in the LMTP client's log file or in the message transaction log file of the "front end" MTA running the LMTP client channel.)

Table 64.1 IMAP error statuses relevant to the MTA

Name	Type	Text+	Meaning
Delivery permanent errors			
IMAP_INVALID_USER	Permanent	Invalid user	The recipient address did not parse into syntactically valid uid, channel part, and optionally domain and/or folder portions
IMAP_QUOTA_EXCEEDED_PERSISTENT	Permanent	Over quota	The user has exceeded their configured quota for more than the configured grace period.
IMAP_MESSAGE_TOO_LARGE	Permanent	Message too large	Message is larger than the user's entire quota (`mailQuota`).
IMAP_MAILBOX_NONEXISTENT	Permanent	Mailbox does not exist	(Recall that these error messages are general IMAP error messages, that may occur in other contexts besides `ims-ms` channel or LMTP channel delivery.) When this error occurs in the context of an `ims-ms` or LMTP channel delivery, the meaning is as follows. Normally, the `ims-ms` channel and LMTP server `tcp_lmtpss*` channel will create a mailbox, if it does not exist. And when delivery to a specific folder is being attempted due to a Sieve `"fileinto"` action, and the folder does not exist, the channel will attempt to create the folder. (If the folder part of the address was not generated by the user's Sieve filter nor world writeable, then the channel will log a General facility, Information level notice to the `imta` file saying "`append_setup` *`path`* `failed, trying INBOX:`" followed by the IMAP_MAILBOX_NONEXISTENT error text. If the folder part was due to the user's Sieve filter but the folder could not be created, then the channel will log a General facility, Error level notice to the `imta` file saying "`mboxlist_createmailbox` *`path`* `failed, trying INBOX:`" with the specific IMAP error text.) But if the partition is bogus, then this error will be returned.
IMAP_MESSAGE_CONTAINSNULL	Permanent	Message contains NUL characters	(A message that has gone through the MTA will not normally cause such an error, as the MTA will normally legalize message content.)
IMAP_MESSAGE_CONTAINSNL	Permanent	Message contains bare newlines	(A message that has gone through the MTA will not normally cause such an error, as the MTA will normally canonicalize message content resulting in proper use of CRLF for line breaks, and encoded newlines in binary content; see the discussion of the `lmtp*` and `smtp*` channel options.)
IMAP_MESSAGE_BADHEADER	Permanent	Message contains invalid header	
IMAP_MESSAGE_NOBLANKLINE	Permanent	Message has no header/body separator	(A message that has gone through the MTA will not normally cause such an error, as the MTA will normally legalize messages one way or another; see the `*headertermination` channel options.)
IMAP_PERMISSION_DENIED	Permanent	Permission denied	
IMAP_CONFIG_ERROR	Permanent	Configuration error	Returned if either LDAP pool code, or the domain map code, cannot be initialized. (Note that normally such initialization is done when the `ims-ms` channel first starts up, and when the LMTP server first starts up, and errors initializing at that time will cause the channel or server to exit. So this is not expected to occur for the `ims-ms` channel and LMTP server `tcp_lmtpss*` channel at this later, message processing, stage.)

IMAP_PROXY_ONLY	Permanent	Server is not configured for local users	
IMAP_WRONG_MAILHOST	Permanent	Mailbox is on a different server	This suggest that either the user's LDAP entry is missing a `mailHost` attribute entirely, or their `mailHost` has been changed (their mailbox has been moved) while old messages were awaiting delivery in the `ims-ms` channel queue area, or that there is an error in the MTA configuration (causing it to attempt to deliver a message on the wrong `mailHost`), or operator error whereby messages have been manually moved (incorrectly) from one host to another.
IMAP_MAILBOX_BADNAME	Permanent	Invalid mailbox name	
	Delivery temporary errors		
IMAP_MAILBOX_LOCKED	Temporary	Mailbox is busy	The mailbox was locked. When this status is encountered, the MTA delivery channel (`ims-ms` or `tcp_lmtpcs*`) first retries 10 times at 100 millisecond intervals (before this error is ever even reported back from the channel). Prior to 8.0, lock attempts were nonblocking; a 1 second time is used in 8.0 or later. If the mailbox remains locked, then the channel gives up on this delivery attempt, generating a "`Q`" record with the "`Mailbox is busy`" reason if `logging` is enabled. And then there is special code in the `ims-ms` channel (and as of MS 7.0 in the LMTP client channel) to ask the Job Controller to retry delivery again very, very soon---with a randomly generated backoff of between one second and two minutes---overriding the normal `*backoff` values. Occasional occurrences of such a temporary error are normal, as for instance if a user is logged in and performing extensive, time-consuming operations on a mailbox; indeed, it is because an occasional such occurrence is "normal" that the delivery channels have special code to handle this (usually transient) error condition by retrying delivery very soon. However, persistent occurrences for the same user might suggest a problem with that user's mailbox. Or, if such errors suddenly start occurring at around the same time for all (or many) users, that could be a suggestion of some more widespread and general problem with the Message Store.
IMAP_MAILBOX_BADFORMAT	Temporary	Mailbox has an invalid format	The mailbox may be corrupted.
IMAP_MAILBOX_NOTSUPPORTED	Temporary	Operation is not supported on mailbox	The mailbox may be corrupted.
IMAP_IOERROR	Temporary	System I/O error. Administrator, check server log for details.	The Message Store may be corrupted, or otherwise inaccessible (*e.g.*, disk not mounted); or the Message Store `store.idx` file may have reached its 2 gigabyte size limit.
IMAP_PARTITION_UNKNOWN	Temporary	Unknown/invalid partition	The user's `mailMessageStore` LDAP attribute does not correspond to a *`partition-name`* defined via a `path` partition option in Unified Configuration (a `store.partition.`*`partition-name`*`.path` configutil parameter in legacy configuration), or the value of that option or configutil parameter does not point to a valid location, or contains invalid characters (only alphanumeric characters are allowed). (If a user does not have a `mailMessageStore` set at all, then the Message Store's default partition, `defaultpartition` Message Store option in Unified Configuration or `store.defaultpartition` configutil parameter in legacy configuration, is assumed, which defaults to `primary`. When the absence of an explicit `mailMessageStore` is causing use of the default partition, then whatever the default partition is, it must then have a valid path specified in the `path` option for that named partition in Unified Configuration (`partititon:`*`whatever`*`.path`) or the `store.partition.`*`whatever`*`.path` configutil parameter; in particular, when `store.defaultpartition` has its default value of `primary`, then the primary partition must have a valid path specified in `partition:primary.path` in Unified Configuration or `store.partition.primary.path` in legacy configuration.)
IMAP_PARTITION_FULL	Temporary	Store partition is full	See also the (new in MS 6.2) configutil parameters `local.store.checkdiskusage` and `local.store.diskusagethreshold` (initially instead named `local.store.diskthreshold` in MS 6.2), or in Unified Configuration the `checkdiskusage` and `diskusagethreshold` Message Store options, which control whether---and when---the store performs checks on its disk space usage. With such checks enabled, the store will "lock" a partition (at which point the IMAP_PARTITION_FULL error will be returned) slightly before the partition is actually filled up. This is a safety measure, to ensure that expunge operations (which need to rewrite the index files) have disk room in which to operate.
IMAP_QUOTA_EXCEEDED	Temporary	Over quota	The recipient has exceeded their configured quota, but for less than the configured grace period.
	Other statuses (not seen by `ims-ms` channels or LMTP channels)[++]		
IMAP_USERFLAG_EXHAUSTED		Too many keywords in mailbox	
IMAP_MAILBOX_EXISTS		Mailbox already exists	When attempting to create a folder, this is a success status
IMAP_MAILBOXLIST_NONEXISTENT		Mailbox list does not exist	
IMAP_INVALID_IDENTIFIER		Invalid identifier	May occur when attempting to set an ACL on a folder
IMAP_INVALID_MSGNO		Invalid message number	
IMAP_QUOTAROOT_NONEXISTENT		Quota root does not exist	

[+] The IMAP error text is localizable. The text shown in this table is merely the default text (which happens to be English).

[++] For completeness, additional IMAP statuses are listed, though they are not normally relevant (do not normally appear) as ims-ms channel or LMTP channel delivery error statuses.

The IMAP delivery temporary error statuses listed in IMAP error statuses relevant to the MTA may appear in the "history" field of messages that failed a delivery attempt; this is visible via, for instance, the history command of the imsimta qm utility, as well as being physically present in the history field in the deferred message files awaiting further delivery attempts in the MTA ims-ms channel disk queue area or tcp_lmtpcs* channel disk queue area. (Should a message eventually bounce due to "timing out", such history information may also be included in the bounce message, as controlled by the history_to_return MTA option.) Furthermore, the temporary error text will be included in the ims-ms channel's "Q" record in the MTA message transaction log file, should such logging be enabled via use of the logging channel option.

The IMAP delivery permanent error statuses shown in IMAP error statuses will appear in the ims-ms channel's or tcp_lmtpcs* channel's "R" or "K" record in the MTA message transaction log file, should such logging be enabled via use of the logging channel option.

As with other types of channel processes, the ims-ms channel processes do not have an infinite life time: after running for a (configurable) while, they shut down (and the Job Controller will start up new ims-ms channel processes, as needed). The Job Controller options max_life_time and max_life_askwork (max_life_age and max_life_conns, respectively, in legacy configuration) and the ims-ms channel's own 60 seconds limit on time to persist while idle, control the timing of the ims-ms channel process "recycling". In addition, an administrator restart of the Job Controller will cause the Job Controller to tell channel processes to shut down. So notices in the imta log file such as the following are part of normal operation:

At Notice level:

product ims_master *version* shutting down

product ims_master *version* starting up

And at Debug level: When the ims-ms channel process' threads have all been idle for 60 seconds or more (or are deadlocked):

idle shutdown

When a thread is exiting due to having no work to do:

T*thread-number* exiting

When the Job Controller is telling a process to shut down due to the process having gotten old:

EOF from job controller

And after an administrator restart of the Job Controller, the following sorts of messages can also be expected in the imta file. At Debug level:

`shutdown received`

When the Job Controller is telling a process to shut down due to an administrator `restart` command:

`exit from job controller` *`mta-status`*

A message (at Warning level) of

`Shutdown timeout, possible deadlock`

however, is an indication that one or more threads in a process attempting to shut down have not been able to finish their work. If this error repeats, it may indicate that a mailbox is corrupted or locked.

In an MTA message transaction log file "`Q`" or "`Z`" record, a reason of "`shutdown`" indicates that a delivery attempt for that recipient is being deferred due to a shutdown in progress---either an administrator's manual shutdown, or a shutdown due to a channel process "timing out".

Chapter 65 Other channels

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 Local channel,
- the `bitbucket` channel,
- the `defaults` and `nodefaults` pseudo-channels,
- 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 `defaults` and `nodefaults` pseudo-channels, which are place-holders rather than actual channels, are discussed under Channel configuration. The `conversion` channel, which has relatively complex configuration, is discussed separately under Message conversions. The other channels listed are discussed in the chapter below.

65.1 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 "l" 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 "l" 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 "l" channel value.

65.2 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 `$DATAROOT/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 enqueuing 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.

65.2.1 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, initial configuration normally creates:

```
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
```

65.3 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 initial configuration.

65.3.1 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.)

65.3.2 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.

65.3.3 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 elapsed[1] (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_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.

65.3.4 Multi-host defragmentation channel operation

The defragment database can optionally be stored on a filesystem accessible to multiple hosts (for instance, over NFS[1]), 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 `config-root/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 amtomount 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 with 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.

65.3.4.1 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.

65.4 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.

65.4.1 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 DATAROOT/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 Sieve "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 Sieve "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" (reinjected 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.

65.5 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. SMTP commands are written to stdout and responses are expected on stdin.

Similarly, test_smtp_slave accepts SMTP commands on stdin and writes responses to stdout.

The 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.

65.6 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.

65.6.1 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 = hold-daemon
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 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
```

65.6.2 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
```

65.6.3 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 threshhold, 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, due to use of a "`hold`" action in a Sieve script or spam/virus filter package action, due to use of a Milter SMFIF_QUARANTINE action (or explicit "`hold;`" in a `QUARANTINE_ACTION` Milter plugin option), or due to a conversion command script exit status of `PMDF__FORCEHOLD`.

When diagnosing the reason why a particular message or messages are `.HELD`, consider the following:write file

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---though *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) or spam/virus filter package action's Sieve scriptlet, or (when using a milter) use of the SMFIF_QUARANTINE milter action, or use of the conversion channel with a script that exits with the status PMDF__FORCEHOLD. Also 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.)

65.7 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 availability of a pipe channel is configurable: the MTA administrator may remove the pipe channel from 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. The forked subprocess then opens the message file on stdin prior to executing the command.

The command can either read the message from stdin or it can 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.

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 `sysexits.h`.

65.7.1 Setting up a pipe channel

There are two steps in setting up a pipe channel:

1. adding the channel definition to the configuration, (note that initial configuration normally generates a pipe channel itself), 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

65.7.1.1 Adding a pipe channel to the configuration

Initial configuration normally configures a pipe channel automatically. In Unified Configuration, that definition would appear as:

```
msconfig> show channel:pipe.*
role.channel:pipe.official_host_name = pipe-daemon
role.channel:pipe.defragment (novalue)
role.channel:pipe.single (novalue)
msconfig> show rewrite.rule * *pipe*
role.rewrite.rule = .pipe-daemon $U%$H.pipe-daemon@pipe-daemon
```

corresponding to the legacy configuration:

```
.pipe-daemon $U%$H.pipe-daemon@pipe-daemon
...rest-of-rewrite-rules...

...start-of-channel-definitions...

pipe single defragment
pipe-daemon
```

But for those wishing to add a pipe channel manually, the following discussion is provided. Note that 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.

The pipe channel runs by default as the Messaging Server user, as specified by the `user` option in `restricted.cnf`. 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 though that usage is deprecated as of MS 8.0; as of MS 8.0, the preferred usage is to use the `pipeuser` option in the `restricted.cnf` file. Note that the argument to `user` channel option 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 define and set delivery methods for pipe channel addressees, as described next.

65.7.1.2 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.

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. In this case addresses of the form "uid%programinfo@pipe-channel" are used. As of Messaging Server 8.0.1.3 recognition of "a%b" form address is enabled by bit 0 (value 1) of the ADDRESS_TYPES pipe channel-specific option. (Bit 0 is set by default.)

Older approaches using the MTA profile database or Pipe channel-specific option file also exist. In these cases addresses of the form "user+domain@pipe-channel" are used. As of MS 8.0.1.3 recognition of "a+b" form addresses is enabled by bit 1 (value 2) of the ADDRESS_TYPES pipe channel-specific option. (Bit 1 is set by default.)

Finally, as of MS 8.0.1.3, when bit 2 (value 4) of the ADDRESS_TYPES pipe channel-specific option is set (which is the default) addresses with a local part not containing a "%" or "+" are handled as if they had a local part of the form: "PIPE-USER%PIPE-CHANNEL_default", where "PIPE-USER" is the value of the `pipeuser` restricted.cnf option and "PIPE-CHANNEL" is the name of the pipe channel (normally "pipe").

65.7.1.2.1 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.

65.7.1.2.1.1 Pipe options

A few options control operation of the Pipe channel 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 of the form "uid%programinfo@pipe-channel", where "uid" is the user's uid (which will be the user the the program runs under) and "programinfo" is the value of the user's `mailProgramDeliveryInfo` attribute (or whatever attribute is named by the `ldap_program_info` MTA option).

The "programinfo" from the address is used as to construct a set of option names which then provide the command to execute, parameters, and permissions. For example, a "foo" program value would check the following options:

```
pipe:foo.command
pipe:foo.params
pipe:foo.perms
```

As of Messaging Server 8.0.1.3, pipe options are also used to handle pipe channel address that do not contain a "%" or "+" charracter. In this case, assuming that the usual pipe channel named "pipe" is used, the following options will be checked:

```
pipe:pipe_default.command
pipe:pipe_default.params
pipe:pipe_default.perms
```

65.7.1.2.1.1.1 `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.

65.7.1.2.1.1.2 `perms` Option

The `perms` Pipe option specifies permissions for the pipe delivery program.

65.7.1.2.1.1.3 `command` Option

The `command` Pipe option specifies the location of the program to execute to perform delivery.

The path can be absolute or relative. The base for relative paths is `DATAROOT`/site-programs. msconfig checks to make sure the specified program exists; it will not allow the option to be set if it doesn't.

65.7.1.2.2 Profile database

The Profile database is located at `IMTA_DATAROOT:db/profiledb`.

Entries in the Profile database are created and managed using the `imsimta profile` utility.

65.7.1.2.3 Profile database entries for pipe channel addressees

Before looking in the 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 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
```

65.7.1.2.4 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.

65.7.1.2.5 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 `CONFIGROOT/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.*, `CONFIGROOT/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` or `%f` 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 all "@", "%", and "+" characters are replaced by underscores. Also 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 (prior to MS 8.0) the `user` channel option on the pipe channel or (preferred approach as of MS 8.0) the `pipeuser` option in `restricted.cnf`, or if no pipe channel user was specified, 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.

65.7.1.2.5.1 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).

65.7.1.2.5.2 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
```

65.7.1.2.5.3 ADDRESS_TYPES (0-7)

New in MS 8.0.1.3, the ADDRESS_TYPES pipe channel option controls what kinds of addresses this pipe channel allows. This is a bit encoded value, with the individual bits defined as follows:

Table 65.1 ADDRESS_TYPES pipe channel option bits

Bit	Value	Meaning
0	1	If set, process addresses of the form a%b.
1	2	If set, process addresses of the form a+b.
2	4	If set, process addresses using default programs.

The default value of this option is 7, meaning all types of processing are enabled.

One use of the ADDRESS_TYPES option is to allow the use of subaddresses with default programs. This can be accomplished by disabling processing of a+b address forms:

```
msconfig> set channel:pipe.options.ADDRESS_TYPES 5
```

65.7.1.2.5.4 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 UNIXplatforms. 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
```

65.7.2 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. (The "@" and any "+" or "%" characters are converted to a "_" when checking unified configuration pipe channel-specific options.)
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.

65.8 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 requeued 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 requeuing 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 MS 6.3, a spam/virus filter package becoming unresponsive also optionally may cause such deferral; see the `spamfilter`*`N`*`_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` channel option) then the reprocessing channel will simply re-enqueue 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-enqueue 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.

65.8.1 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.

65.8.1.1 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 66 SMS options

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.

66.1 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.

66.1.1 `enable` Option Under sms_gateway

The `enable` SMS gateway option, `sms_gateway.enable` (Unified Configuration) or `local.smsgateway.enable` (legacy configuration), enables the SMS service on `start-msg` startup.

66.1.2 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 66.1 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.

66.1.3 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.

66.1.4 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_DATAROOT: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.

66.1.5 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.

66.1.6 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.

66.1.7 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.

66.1.8 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.

66.1.9 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.

66.1.10 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.

66.1.11 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.

66.1.12 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.

66.1.13 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.

66.2 SMS gateway_profile options

66.2.1 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.

66.2.2 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.

66.2.3 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.

66.2.4 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.

66.2.5 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.

66.2.6 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.

66.2.7 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.

66.2.8 `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".

66.2.9 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.

66.2.10 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.

66.2.11 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.

66.2.12 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 66.2 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.

66.2.13 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 66.3 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.

66.3 SMS smpp_relay options

There are a number of options under smpp_relay.

66.3.1 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.

66.3.2 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).

66.3.3 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).

66.3.4 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).

66.3.5 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.

66.3.6 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.

66.3.7 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.

66.3.8 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.

66.3.9 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).

66.3.10 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).

66.3.11 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.

66.4 SMS smpp_server options

There are a number of options under smpp_server.

66.4.1 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.

66.4.2 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 66.4 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 various server_port options for further information.

66.4.3 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.

66.4.4 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 66.5 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.

66.4.5 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.

66.4.6 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.

66.4.7 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.

66.4.8 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).

66.4.9 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).

66.4.10 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).

66.4.11 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.

66.4.12 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.

66.4.13 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.

66.4.14 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.

66.4.15 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 67 Message capture

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, or use of the `clonehosts` channel option to generate "clone" copies of messages to an additional destination, 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 Capturing messages via Sieve scripts. 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 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 MS 6.3) Integration with the AXS:One archive facility, used to generate message copies that will be archived by AXS:One, discussed in AXS:One archive integration. 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 capt ured 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.

67.1 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). (The new-in-MS-8.0 `clonehosts` channel option provides a different facility, with different trade-offs, that can also be suitable for certain "message replay" purposes.) The `MESSAGE-SAVE-COPY` 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.

67.1.1 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`* `$Y`*`result-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:

`$Y$G`*`copy-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 MS 6.3, such a task would typically require use of a special delivery channel -- see Additional ims-ms channels -- or in 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.

67.1.2 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.

```
# 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
```

67.2 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.)

67.3 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 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.

67.4 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 ESMTP id <0KRH00A3QAO42T10@host.domain.com>
 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 subpoena1-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: subpoena1-on-user1@domain.com                            (4)
Message-id: <0IH400G08EULG800@ketu.west.sun.com>
MIME-version: 1.0
Original-recipient: rfc822;subpoena1-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 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)                  (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 `subpoena1-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 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 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 overriden 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
 (buil
t Sep  1 2009)) with ESMTP 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 enqueuing 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_xWvHuuTuAMSgGf6g0CDWjg)" (8)


--Boundary_(ID_xWvHuuTuAMSgGf6g0CDWjg)
Content-type: text/plain                                          (9)

Sender: <user1@domain.com>                                        (10)
Message-ID: <0IH400G04EU3G800@host.domain.com>                    (11)
Recipients:
 <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 7.3-11.01 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_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 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.

67.5 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, preferrably 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.

67.5.1 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.

67.5.2 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.

67.5.3 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.

67.5.3.1 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`.

67.5.3.2 MTA support for AXS:One archiving

(New 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 "identifer" 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_optin`*`N`* MTA options, the `*spamfilter`*`N`*`optin` channel options, and the (new in MS 6.3) `ldap_source_optin`*`N`* 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.

67.5.3.3 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 `spamfilter`*`N`*`_config_file` and `spamfilter`*`N`*`_library` MTA options. For some value of *N*, the `spamfilter`*`N`*`_library` MTA option must be set to point to the `libarch.so` module, while the `spamfilter`*`N`*`_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.

67.5.3.3.1 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., CONFIGROOT*/archive.dat) file:

```
STYLE=1
DIRECTORY=/opt/SUNWmsgsr/archive/
! For compliance mode (legal archiving requirements compliance):
IDSUFFIX=-0000MD500
```

Chapter 68 Monitoring the MTA

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.

68.1 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.

The mail.log and connection.log entries may optionally be duplicated to syslog via the log_messages_syslog and log_connections_syslog options.

68.1.1 Managing the MTA transaction log files

When the logging channel option is enabled, all MTA message transaction log entries are made to the file mail.log_current in the MTA log directory, of MS 7.0.5 *DATAROOT*/log/mail.log_current; (prior to MS 7.0, the MTA message transaction log file was located via the imta_primary_log MTA Tailor option). If connection logging is enabled via the log_connection MTA option, connection transaction log entries are also by default written to the mail.log_current file, but if the MTA option separate_connection_log=1 has been set, then the connection transaction log entries will instead be written to the connection.log_current file.

The message return_job, which the Scheduler is typically configured to run every night shortly after midnight, appends any existing mail.log_yesterday to the cumulative log file, mail.log, renames the current mail.log_current file to mail.log_yesterday, and then begins a new mail.log_current file. It also performs the analogous operations for any connection.log* files. (In older versions of the MTA, the names and location of these log files were controlled by the imta_primary_log, imta_secondary_log, and imta_tertiary_log MTA Tailor options; as of MS 7.0.5, these names are not configurable and the location is derived from the SERVERROOT value.) (Note that the return_split_period MTA option can modify the frequency of such transaction log file "roll over".)

Note that the MTA itself by default never does anything to the cumulative mail.log file; it is up to each site to manage that log file however they choose, whether by periodically saving it to backup, deleting it, truncating it, or the like.

When considering how to manage the log files, note that the MTA periodic return_job will execute a site-supplied *DATAROOT*/site-programs/bin/daily_cleanup shell

script, if one exists. Thus some sites may choose to supply their own `daily_cleanup` that, for instance, renames the old `mail.log` file once a week (or once a month), *etc.* (Note that the `return_cleanup_period` MTA option can modify the frequency at which such "cleanup" is performed.)

68.1.2 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.
          (1)         (2)             (3)      (4)(5)    (6)         (7)                        (8)
```

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 68.1 Message logging entry action type codes

Type	Modifiers	Description
		General
B	E, P, A, S, U, B, C	(New in MS 6.2) Unrecognized/invalid SMTP command. Logging limit set by `MAX_B_ENTRIES` TCP/IP channel-specific option.
D	E, L, A, S, U, B, C	Successful dequeue
E	E, P, A, S, U, B, C, W, T	Enqueue. An "E" or null modifier will be present to indicate whether the enqueue was with EHLO or HELO, respectively.
H	E, P, A, S, U, B, C	(New in MS 8.1) VRFY/EXPN command. Logging limit set by `MAX_H_ENTRIES` TCP/IP channel-specific option.
J	E, P, A, S, U, B, C, W, T, R	Rejection of attempted enqueue (rejection by slave channel program). Logging limit set by `MAX_J_ENTRIES` TCP/IP channel-specific option,
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: • 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 • 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 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. Logging limit set by `MAX_J_ENTRIES` TCP/IP channel-specific option, As of MS 7.0, J records are 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.
	8	(New in MS 8.0) UTF-8 message transfer (UTF8SMTP) successfully used
	W	(New in MS 8.1.0.1) Lines longer than 1000 characters were found and wrapped
	T	(New in MS 8.1.0.1) Lines longer than 1000 characters were found and truncated
	R	(New in MS 8.1.0.1) Lines longer than 1000 characters were found, causing the message to be rejected

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) (log_node) The name of the node on which the channel process is running.
- (11) (log_process) 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) (log_notary) The NOTARY (delivery receipt request) flags for the message, expressed as an integer.
- (13) (log_filename) The file name in the MTA queue area.
- (14) (log_envelope_id) 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) (log_message_id) The message id.
- (16) (log_username) 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) (log_connection) 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 enqueuing 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) (`log_sensitivity`) 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) (`log_priority`) 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 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 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 MTA, especially as of 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 "I" 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", "Y" records may include the "time attempting to open a connection for an attempt that failed", and "C" records may include the total time the connection was open as a final field. This appears as a ct attribute in XML format logs.

Table 68.2 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 log_connection MTA option is set (or overridden by the LOG_CONNECTION TCP/IP-channel-specific option), and the $T flag is used in a PORT_ACCESS rejection entry
U		(New in MS 6.2) Authentication attempt (SMTP AUTH use), whether successful or failed; logged if bit 5 (value 32) of the log_connection 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 log_connection MTA option is set (or overridden by the LOG_CONNECTION 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.

[1]The 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.

68.1.3 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.

68.1.3.1 LOG_ACTION operation

The LOG_ACTION mapping table is probed each time a message transaction or connection transaction log entry is written. At present 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.

68.1.3.2 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; (*i.e.*, the ac attribute's value in XML format MTA transaction logging). 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|remote-mta|isc-status
|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_remote_mta, log_isc_status, 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*|*extra*|*adminser*|*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_conversion_tag (especially relevant for open/close SMTP 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.

Mapping input flags are described in the following table.

Table 68.3 LOG_ACTION mapping input flags

Flag	Description
$|	(New in MS 8.0.1.2) Set if one or more of the input strings contains a vertical bar character. (A vertical bar in an input string has the potential of being misinterpreted as a delimiter.)
$R	(New in MS 8.0.1.2) Set if the source channel is an "internal" channel and the MTA is operating in reprocessing mode.
$S	(New in MS 8.0.2.3) Set if the per-entry save flag is set and the entry is going to be written to the log file; clear otherwise.

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 68.4 LOG_ACTION mapping output flags

Flag	Description
$F	Disable writing this entry to the transaction log file
$N	Disable writing this entry to the transaction log file
$+|	Causes any syslog message produced by $< or $> to consume all remaining arguments as well as including the vertical bars themselves.
Output flags with arguments, in argument reading order[1]	
$T*probe-tag*	(New in MS 8.0) (Only takes effect for message transaction probes, not connection transaction probes.) Prefix subsequent *_ACCESS mapping table probes with the tag *probe-tag*. The last such tag may optionally be prepended to the AUTH_REWRITE mapping table probe.
$D*N*	(New in MS 8.0.1.2) Delay the session after logging operation is complete. The delay is controlled by the signed integer parameter value *N*. Positive values of N specify the number of centiseconds to delay; if multiple log records are being written as part of a transaction the value associated with the last matching probe is used. Negative values of N are cumulative; the absolute values of matching probes are added together to produce the number of centiseconds to delay.
$<*string*	Send *string* to syslog (UNIX) if probe matches; see also the sndopr_priority MTA option.
$>*string*	Send *string* to syslog (UNIX) if access is rejected; see also the sndopr_priority MTA option.

[1] To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

68.1.3.3 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.

68.1.3.3.1 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
```

68.1.3.3.2 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> set sndopr_prefix ""
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
!
! Eliminate the syslog prefix
!
SNDOPR_PREFIX=
```

(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.

68.1.3.3.3 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
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)
```

68.1.3.3.4 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.)

68.1.3.3.5 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


LOG_ACTION

! source-chan|dest-chan|D|size|env-from|orig-env-to|env-to|filename|queue-time
!
  *|*|D*|*|*|*|*@*|ZZ*|*   $CDEQUEUEDOMAINTIME|$7|$9
!
! Now probing with just DEQUEUEDOMAINTIME|domain|queue-time
!
  DEQUEUEDOMAINTIME|*|*    $CFASTDOMAIN|$`integer($1)<=60'|$0|$1
!
! Now the probe will be FASTDOMAIN|1|domain|queue-time for queue-time<=60s
! or FASTDOMAIN|0|domain|queue-time for queue-time>60
!
  FASTDOMAIN|1|*|*         \
$[IMTA_LIB:check_metermaid.so,remove,slow_delivery,$0]
  FASTDOMAIN|0|*|*         $CSLOWDOMAIN|$`integer($1)>=300'|$0|$1
!
! Now the probe will be SLOWDOMAIN|1|domain|queue-time for queue-time>=300s
! or SLOWDOMAIN|0|domain|queue-time for 60s<queue-time<300s
!
  SLOWDOMAIN|1|*|*         \
$[IMTA_LIB:check_metermaid.so,throttle,slow_delivery,$0]$<LOGACTION,$ \
Domain$ $0$ deliveries$ slow
```

The above example uses the mapping table expression substitution syntax, `$`expression'`, to perform a test on the value of the `queue-time` (time in queue) field: such an expression will evaluate to 1 if true, 0 if false, so such an expression substitution inserts the string `1` if the numeric test was true or inserts the string `0` if the numeric test was false.

68.1.3.3.6 Blocking submissions of local senders who may be spammers

(Note that this example uses the MeterMaid "test" routine, new in Messaging Server 7.0 update 2.) A different type of use of `LOG_ACTION` is to update a MeterMaid table with information relating to message dequeue, which a message enqueue access mapping table can then consult. That is, `LOG_ACTION` with MeterMaid can provide a way for decisions on allowing message enqueue to reference historical data regarding message dequeue. For instance, a local user

whose outbound to the Internet messages suffer a high number of rejections of bad addresses by remote destinations may raise the suspicion that that user is attempting to send spam using a poor quality recipient list; a site may wish to deny that account further submissions that day.

For any such access restrictions on senders, it is always strongly advisable to *require* local users to authenticate (use SMTP AUTH) in order to submit messages, and then to perform the access checks using the *authenticated* submission address. Because the SMTP envelope From is easy to forge in the base SMTP protocol, attempts to enforce access restrictions based solely upon envelope From are regrettably likely to unintentionally *encourage* (or at least motivate) users to forge their envelope From as a way to bypass the access restrictions. Thus poorly considered access restrictions on senders can cause more harm than good, by motivating users to obscure their identity to evade the restrictions. So this example will assume that a security-conscious configuration, where users are required to use SMTP AUTH to submit, is already in place, so that authentication information is available both for logging via the LOG_USERNAME MTA option, and for performing an access check from the FROM_ACCESS mapping table.

So with MTA options settings that include (legacy configuration style):

```
! 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.

68.1.3.3.7 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.

68.1.3.3.8 Delay after bad username and password specified in SUBMIT

Given the following MTA option settings:

```
LOG_CONNECTION=515
LOG_USERNAME=3
LOG_DIAGNOSTICS=3
```

And mapping:

```
LOG_ACTION

  tcp_*|+|U|*|535*      $D300
```

A three second delay would follow any failed authentication attempt.

68.2 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.

68.2.1 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 68.5 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 68.6 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 then 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)
ConvSuccessed	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 `local.snmp.directoryscanconfigutil` 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.

68.2.2 Purpose and design 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 in 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.

68.2.3 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.

68.2.4 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.

68.2.4.1 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 68.7 Supported MIB variables

applicationTable variables			
Variable name	OID	Syntax	Value
applName	mib-2.27.1.1.2	SnmpAdminString	*instance-nameservice-name* on *host-name*
applVersion	mib-2.27.1.1.4	SnmpAdminString	*major-version*. *minor-version***or***major-version*. *minor-version* Patch *patch-version*
applDirectoryName	mib-2.27.1.1.3	SnmpAdminString	NULL

applUpTime	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
applOperStatus	mib-2.27.1.1.6	Integer	1 (up) **or** 2 (down)
applLastChange	mib-2.27.1.1.7	TimeStamp	Time (in hundredths of seconds) since the Job Controller's pidfile was modified
applInboundAssociations	mib-2.27.1.1.8	Gauge32	Sum over all mtaGroups (channels) of the mtaGroupInboundAssociations
applOutboundAssociations	mib-2.27.1.1.9	Gauge32	Sum over all mtaGroups (channels) of the mtaGroupOutboundAssociations
applAccumulatedInboundAssociations	mib-2.27.1.1.10	Counter32	Sum over all mtaGroups (channels) of the mtaGroupAccumulatedInboundAssociations
applAccumulatedOutboundAssociations	mib-2.27.1.1.11	Counter32	Sum over all mtaGroups (channels) of the mtaGroupAccumulatedOutboundAssociations
applLastInboundActivity	mib-2.27.1.1.12	TimeStamp	Most recent of the mtaGroupLastInboundActivity values
applLastOutboundActivity	mib-2.27.1.1.13	TimeStamp	Most recent of the mtaGroupLastInboundActivity values
applRejectedInboundAssociations	mib-2.27.1.1.14	Counter32	Sum over all mtaGroups (channels) of the mtaGroupRejectedInboundAssociations
applFailedOutboundAssociations	mib-2.27.1.1.15	Counter32	Sum over all mtaGroups (channels) of the mtaGroupFailedOutboundAssociations
applDescription	mib-2.27.1.1.16	SnmpAdminString	*ims-nameapplVersion-value*
applURL	mib-2.27.1.1.17	URLString	NULL
mtaTable variables			
Variable name	OID	Syntax	Derived from MTA counter
mtaReceivedMessages	mib-2.28.1.1.1	Counter32	received_messages
mtaStoredMessages	mib-2.28.1.1.2	Gauge32	stored_messages
mtaTransmittedMessages	mib-2.28.1.1.3	Counter32	delivered_messages
mtaReceivedVolume	mib-2.28.1.1.4	Counter32	received_volume (converted, as necessary, to Kbytes)
mtaStoredVolume	mib-2.28.1.1.5	Gauge32	stored_volume (converted, as necessary, to Kbytes)
mtaTransmittedVolume	mib-2.28.1.1.6	Counter32	delivered_volume (converted, as necessary, to Kbytes)
mtaReceivedRecipients	mib-2.28.1.1.7	Counter32	received_recipients
mtaStoredRecipients	mib-2.28.1.1.8	Gauge32	stored_receipients
mtaTransmittedRecipients	mib-2.28.1.1.9	Counter32	delivered_recipients
mtaSuccessfulConvertedMessages	mib-2.28.1.1.10	Counter32	Sum over all mtaGroups (channels) of mtaGroupSuccessfulConvertedMessages; since all non-`conversion*` channels have 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-2.2.28.1.17	TimeInterval	Channel counter lastinbound
mtaGroupLastOutboundActivity	mib-2.2.28.1.18	TimeInterval	Channel counter lastoutbound
mtaGroupInboundRejectionReason	mib-2.2.28.1.21	SnmpAdminString	NULL
mtaGroupOutboundConnectFailureReason	mib-2.2.28.1.22	SnmpAdminString	NULL
mtaGroupScheduledRetry	mib-2.2.28.1.23	TimeInterval	0
mtaGroupMailProtocol	mib-2.2.28.1.24	OID	*smtp-oid*, *i.e.*, 9.1.3.6.1.2.1.27.4.25
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	*instance-name* MTA *channel-name* channel
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-2.2.28.1.34	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<=n<=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` utility's `counters` command. Refer to the discussion of MTA channel counters for further information on the MTA channel counters.

Table 68.8 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 69 MTA performance tuning

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.

69.1 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. On an LMTP back end system, see instead the `BUFFER_SIZE` TCP/IP-channel-specific 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` TCP/IP-channel-specific 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 competion among separate outgoing TCP/IP channels.

- Disabling the MTA's SMTP server creation and use of `*.data-failed` files by setting the TCP/IP-channel-specific option `REUSE_TIMED_OUT_TRANSFERS=0` can provide a noticable 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.

- Use of LMTP back ends allows separation of MTA systems from back end Message Store systems, so that individual systems may be dedicated to one type of functionality or the other. Configuration of MTA LMTP client channels can benefit both from the sorts of considerations for `ims-ms` channels (regarding Message Store delivery) and from the sorts of considerations for TCP/IP channels (regarding the network connectivity). Consider using a separate Job Controller `pool` or `pools` for the LMTP client channel(s). When there are multiple back end LMTP systems, note that there are tradeoffs between using one channel to deliver to all back ends, *vs*. creating separate channels to deliver to each back end.

- 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".

69.2 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.

By using symbolic links under the MTA's queue and log directories, you can redirect where the MTA keeps its store-and-forward message queues and log files. The MTA's command, executable, and configuration directories can also be separated if absolutely necessary.

Note that MTA debugging, which can be enabled for various components, and at various levels, (see for instance, the `job_controller.debug` and `dispatcher.debug` options for the Dispatcher and Job Controller, respectively, and the `master_debug` and `slave_debug` channel options for channels) is in some cases potentially very verbose and voluminous.

MTA debugging is distinct from MTA transaction logging (which is, however, also written to the MTA log directory). Debugging is intended for short-term use to track down problems. Leaving debugging enabled unnecessarily for extended periods may make the MTA log directory "hotter" than desirable for the long term.

The location for MTA temporary files can also be moved. The `tmpdir` MTA option (formerly `imta_tmp` in the MTA Tailor file) controls the location of temporary unnamed files (such as those used to buffer incoming large SMTP messages or incoming large messages submitted by local users, and to buffer *all* incoming LMTP messages); it also controls the location of temporary named files (such as those used by the conversion channel). `mta.tmpdir` defaults to `/tmp/`; note than on Linux, it tends to be preferable to set it explicitly to `/dev/shm/`. Note that if explicitly defining `tmpdir` (and intending to support message submissions directly from command line utilities by users logged onto the MTA system itself) it is important to point it to a device on which any user may create files.

By default, the messages for a given channel are stored in a single, channel-specific directory under the MTA queue directory, *`DATAROOT`*`/queue/`. File system performance degrades rapidly for directories with more than a couple thousand files; this can present a problem for channels which see heavy message traffic --- especially when the network associated with that channel is down and messages begin to queue up. Use the `subdirs` channel option to indicate that a channel should uniformly spread its messages across several subdirectories. For Internet sites with heavy traffic loads, this should be done for their outgoing TCP/IP channel, usually `tcp_local`.

MTA options such as `osync` and `buffer_size` may affect file system I/O performance and hence impact MTA performance.

UNIX sites may consider whether for their use it is acceptable to set the MTA option `fsync=0`. Doing so improves performance, but at the cost that if a UNIX (or NT) system crashes at just the wrong moment, messages not yet synched to disk could be lost.

Sites running on Solaris, and especially on Solaris 10 (which has a high default `rlim_fd_max` value), if running a version of the MTA prior to MS 6.4, should take steps to check and ensure that a high system maximum file descriptor limit is not unduly burdening the Job Controller and the channel delivery jobs that it initiates; see System parameters on Solaris.

69.3 System parameters on Solaris

69.3.1 For the Dispatcher:

As of Solaris 11, the `siffp_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 accomodate 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 csh command

```
# limit
```

or the ksh command (see the data result)

```
# ulimit -a
```

or the utility

```
# sysdef
```

69.3.2 For the Job Controller:

As of 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 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 70 Restricting information emitted

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.

70.1 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.

70.2 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 repliable 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. New in MS 7.0.5.37, the `forcedreceivedfrom` channel option

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 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 sytems 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 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.

70.3 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 71 MTA command line utilities

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
```

Generally, use of these utilities requires superuser privileges, or being logged in as the MTA user (see the `user` option in the `restricted.cnf` file).

71.1 cache -change utility

Change Job Controller option effective values for the currently running Job Controller process.

71.1.1 Syntax

```
imsimta cache -change
```

Table 71.1 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`

71.1.2 Parameters

None.

71.1.3 Description

(New in 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`).

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 channel 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.

71.1.4 Switches

71.1.4.1 -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.

71.1.4.2 -debug=*n*

Set a debug level for Job Controller operation. -global must be specified in order to use -debug.

71.1.4.3 -global

Set certain global Job Controller options.

71.1.4.4 -inorder_rebuild, -noinorder_rebuild

-inorder_rebuild directs the Job Controller to rebuild the message list in order. The -global switch must be specified in order to use -inorder_rebuild. -noinorder_rebuild (the default) means to not rebuild the queues in order: to simply insert messages into the queue cache in whatever order they are encountered while scanning the queue area.

71.1.4.5 -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.

71.1.4.6 -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.

71.1.4.7 -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 the value for a particular channel, or with the -template switch, to set the value for a class of channels.

71.1.4.8 -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 the value for a particular channel, or with the -template switch, to set the value for a class of channels.

71.1.4.9 -parallel_rebuild=*n*

Override the value of the rebuild_parallel_channels Job Controller option. -global must be specified in order to use -parallel_rebuild.

71.1.4.10 -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_*.

71.1.4.11 -thread_depth=*n*

The -channel=*name* switch must be specified in order to use -thread_depth; override the effective threaddepth for the specified channel.

71.1.5 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.

71.2 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.

71.2.1 Syntax

```
imsimta cache -synchronize
```

Table 71.2 imsimta cache -synchronize Command Switches

Switch	Default
`-debug=`*n*	`-nodebug`
`-recipient`	`-recipient`

71.2.2 Parameters

None.

71.2.3 Description

The `imsimta cache -synchronize` utility tells the MTA Job Controller to re-scan the queue disk area, *DATAROOT*`/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.

71.2.4 Switches

71.2.4.1 `-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`.

71.2.4.2 `-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.

71.2.5 Examples

To synchronize the queue cache, for instance after renaming a message file, issue the UNIX command

```
# imsimta cache -synchronize
```

71.3 cache -view utility

View the current entries for a channel in the MTA queue cache database.

71.3.1 Syntax

```
imsimta cache -view [channel-name]
```

71.3.1.1 Restrictions

Must have superuser privileges or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.3.2 Parameters

71.3.2.1 *channel-name*

Optional parameter specifying the name of the channel for which to show entries. If no channel name is specified, all entries in the queue cache database will be shown.

71.3.3 Description

The `imsimta cache -view` utility shows the current entries in the MTA queue cache database for a channel.

71.3.4 Examples

This example shows checking the queue cache database for entries for the `ims-ms` channel and finding one such entry:

```
# imsimta cache -view ims-ms
                channel : ims-ms
              file name : /opt/sun/comms/messaging64/data/queue/ims-ms/010/ZYi041V0jMRS0.00
            subdirectory : 10
            enqueue time : 21-Sep-2015:10:51:36.00
   last delivery attempt : 21-Sep-2015:10:52:15.00
     processing priority : 3
      destination system :  ims-ms
         recipient count : 1
            message size : 1 (kbyte)
                   owner :
         expiration date : 18-Jan-2038:19:14:07.00
                due date : No due date recorded
-----------------------------------------------------
```

71.4 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.

71.4.1 Syntax

```
imsimta cache -walk
```

Table 71.3 imsimta cache -walk Command Switches

Switch	Default
`-debug=`*n*	`-nodebug`

71.4.2 Parameters

None.

71.4.3 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.

71.4.4 Switches

71.4.4.1 `-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`.

71.4.5 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
```

71.5 calc utility

The calc utility provides an interface to the MTA's internal expression parser and evaluator. Additional function "sets" can be added by specifying appropriate switches, e.g., the -mm switch provides access to the Sieve function set.

71.5.1 Syntax

```
imsimta calc [expression]
```

Table 71.4 imsimta calc Command Switches

Switch	Default
-input=*filename*	*None*
-output=*filename*	*None*
-statement=*n*	-statement=1
-debug[=*n*]	-debug=4
-uav=*n*	-uav=1
-multiple	-multiple
-mm	*See text*
-message=*filename*	*None*
-symbols	-nosymbols
-from	-from=""
-required	-required
-to	*None*
-system	-nosystem
-mtpriority=*n*	-mtpriority=0
-source=*source-channel-name*	-source=l
-rsecret=*recall-secret*	

71.5.1.1 Restrictions

None.

71.5.1.2 Prompts

calc>*expression*

71.5.2 Parameters

71.5.2.1 *expression*

The expression to evaluate.

71.5.3 Description

The `calc` utility is a calculator, for performing integer and string operations. See Operators in Order of Precedence for a list of basic arithmetic and symbolic operators available, and Symbol table functions for a list of functions available.

Variables can be created, and values assigned to them, using the "=" assignment operator, *e.g.*,

```
calc> a = 3
```

Use Ctrl-D to exit.

In addition to its use for performing simple calculations, the `imsimta calc` utility may also be helpful for testing planned use of calculations in Sieve filters, or in expressions inside MTA mapping tables. Compared with the `imsimta test -expression` utility, the `calc` utility is simpler for testing the fundamentals of mathematical calculation or string manipulation, but does not test, for instance, Sieve filter-specific syntax.

71.5.4 Switches

71.5.4.1 `-debug=`*`n`*, `-nodebug` (default)

The `-debug` switch may be used to enable debug output. Specifying `-debug` is equivalent to specifying `-debug=4`.

71.5.4.2 `-from=`*`address`*, `-to=`*`address`*

(`-from` new in MS 6.2; `-to` new in MS 6.3.) The `-mm` and `-message` switches must be used in order to use `-from` or `-to`. With `-from`, specify the envelope From address (for instance, for purposes of Sieve filter evaluation). If not specified, the MTA's default postmaster return address is assumed. If `-message` is not specified, then `-from` has no effect. With `-to`, specify the envelope To address.

71.5.4.3 `-input=`*`filename`*, `-noinput` (default)

The `-input` switch may be used to specify a file from which to read the expressions to be evaluated.

71.5.4.4 `-message=`*`filename`*, `-nomessage` (default)

Initialize the MTA as if for message submission of the specified message. The `-mm` switch must be used in order to use `-message`. When `-message` is used, the `-from` switch and `-to` switch also become available.

71.5.4.5 `-mm`, `-nomm` (default)

If specified, loads the Sieve function set. This causes an MTA initialization call to be made. Use of any of the `-message`, `-required`, or `-system` switches requires that `-mm` also be specified. The `-mm` and `-sy` switches are mutually exclusive (they cannot be used with each other).

71.5.4.6 `-mtpriority=`*`n`*, `-nomtpriority`

(New in MS 8.0) `-mtpriority` takes a required integer argument specifying the initial MT-PRIORITY value. It may only be specified when `-mm` has also been specified.

71.5.4.7 `-multiple` (default), `-nomultiple`

The default, `-multiple`, allows evaluating multiple, comma-separated statements per physical line. Specifying `-nomultiple` disables this, so that only the first result (from the physical line) is returned.

71.5.4.8 `-output=`*`filename`*, `-nooutput` (default)

The `-output` switch may be used to specify a file to which to write the output of the calculator.

71.5.4.9 `-required` (default), `-norequired`

(New in MS 6.2.) The `-mm` switch must be used in order to use `-required`.

71.5.4.10 `-rsecret=`*`recall-secret`*

(New in MS 8.0) Specify the recall secret.

71.5.4.11 `-source=`*`source-channel-name`*

(New in MS 8.0) Specify the source channel; the default is `l` (the "L"ocal channel).

71.5.4.12 `-statement=`*`n`*

n is a bit-encoded integer. The default is `-statement=1`. But note that specifying `-mm` forces setting bit 9 (value 512), overriding whether or not `-statement` set that bit, to allow bracketed lists.

See the `test -expression` utility's description of Statement parsing flags for details on the meaning of the bits for this switch's value.

To test Sieve scripts using the `"loop"` construct or (new in MS 8.0) `"sub"` construct, note that `-statement=3` must be specified.

71.5.4.13 `-symbols`, `-nosymbols` (default)

The `-symbols` switch, if specified, enables references to environment variables. `-nosymbols` is the default.

71.5.4.14 `-system`, `-nosystem` (default)

(New in MS 7.0.5) The `-mm` switch must be used in order to use `-system`. If `-system` is specified with `-mm`, then the Sieve is treated as a system-level Sieve; if not, then it is treated as a user-level Sieve.

71.5.4.15 `-uav=`*`n`*

The `-uav` switch controls the interpretation of unassigned variables. The default is `-uav=1`.

Table 71.5 calc utility's -uav switch values

Value	Usage
0	Variables must be predefined; variable creation not allowed
1	Assignment defines variable; no default value
2	Define variable upon first reference; default value ""
3	Define variable upon first reference except in modify operations; default value 0
4	(New in MS 8.0/patch to MS 7.0u5) Same as value 1 (assignment defines variable; no default value), with the difference (appropriate for Sieve usage) don't create a symbol table at parse time

71.6 chbuild utility

Compile the MTA character set conversion tables, and `tlds.txt` file.

71.6.1 Syntax

```
imsimta chbuild
```

Table 71.6 imsimta chbuild Command Switches

Switch	Default
`-image_file=`*`file-spec`*	`-image_file=IMTA_CHARSET_DATA`
`-maximum`	`-nomaximum`
`-option_file=`*`file-spec`*	`-nooption_file`
`-remove`	*None*
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`

71.6.1.1 Restrictions

Must have superuser privileges, or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.6.2 Parameters

None.

71.6.3 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 `tlds.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 `tlds.txt` file has been fetched from IANA. (Note that to get updates to `tlds.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, *config-root*`/advanced/charset_data` and *server-root*`/lib/option_charset.dat`, where *server-root* is the product install directory or the value of the `SERVERROOT` environment variable.

71.6.4 Switches

71.6.4.1 -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) *CONFIGROOT*/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.

71.6.4.2 -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) *CONFIGROOT*/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.

71.6.4.3 -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 *CONFIGROOT*/advanced/option_charset.dat, then *CONFIGROOT*/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 *CONFIGROOT*/advanced (or at least to *CONFIGROOT*) 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 *CONFIGROOT*/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.

71.6.4.4 `-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) *CONFIGROOT*`/advanced/charset_data`.

71.6.4.5 `-sizes`, `-nosizes` (default)

The `-sizes` switch instructs `imsimta chbuild` to output information on the sizes of the uncompiled character set tables.

71.6.4.6 `-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.

71.6.5 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 CONFIGROOT/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
```

71.7 clbuild utility

Compile an MTA command definition file and generate an image file (suitable for memory mapping by MTA processes).

71.7.1 Syntax

```
imsimta clbuild cld-file-spec
```

Table 71.7 imsimta clbuild Command Switches

Switch	Default
`-debug`	`-nodebug`
`-image_file=`*`file-spec`*	`-noimage_file`
`-maximum`	`-nomaximum`
`-option_file=`*`file-spec`*	`-nooption_file`
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`

71.7.1.1 Restrictions

Must have superuser privileges or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.7.2 Parameters

71.7.2.1 *cld-file-spec*

The file specification of an MTA command definition file to read as input; *e.g.,* on UNIX `SERVERROOT/lib/pmdf.cld`.

71.7.3 Description

The `imsimta clbuild` utility compiles a command line definition file and generates a binary file (suitable for memory mapping by MTA processes).

The MTA ships with any pre-compiled command line definition files it needs so it is not normally necessary to run this utility.

71.7.4 Switches

71.7.4.1 -debug, -nodebug (default)

The `-debug` switch causes `imsimta clbuild` to output debug information regarding its operation.

71.7.4.2 -image_file=*file-spec*, -noimage_file (default)

By default, imsimta clbuild does not produce a compiled command definition image file. In order to produce a compiled command definition file, the file to produce must be specified using the -image_file switch.

71.7.4.3 -maximum, -nomaximum (default)

The file SERVERROOT/lib/maximum_command.dat is read when -maximum is specified. This file specifies near maximum table sizes but does not change any other command option file parameter settings. Only use this 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 command definition image that is produced by -maximum, but it does make sense to use -maximum to get past size restrictions in order to build a properly sized command option file so that a properly sized command definition image can be built with a subsequent imsimta clbuild invocation.

71.7.4.4 -option_file[=*file-spec*], -nooption_file (default)

imsimta clbuild can optionally produce a command option file that contains correct table sizes to hold the command definitions which were just compiled (plus a little room for growth). The -option_file switch causes this file to read as input and a new such option file created as output. If -option_file is specified with no value, then the file written will have the same name as the input command definition file, but with the file extension .cop; for instance, if the file SERVERROOT/lib/pmdf.cld was the input parameter, then the default name for the output command option file would be SERVERROOT/lib/pmdf.cop. If the -nooption_file switch is specified (the default), then no option file will be output. Note that use of the -maximum switch causes imsimta clbuild to read options from maximum_command.dat in addition to any command option file. This file specifies near maximum table sizes. Only use this switch if the current table sizes are inadequate, and only use it to create a new 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.

71.7.4.5 -remove

Remove an existant compiled command definition image.

71.7.4.6 -sizes, -nosizes (default)

The -sizes switch instructs imsimta clbuild to output information on the sizes of the uncompiled command definitions.

71.7.4.7 -statistics, -nostatistics (default)

The -statistics switch instructs imsimta clbuild to output information on the compiled command definition image. 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.

71.7.5 Examples

The standard command used to compile the basic MTA command definition file on UNIX is:

```
# imsimta clbuild -option -image=IMTA_TABLE:advanced/command_data IMTA_LIB:pmdf.cld
```

To remove the compiled version of the basic MTA command definitions, use

```
# imsimta clbuild -remove IMTA_TABLE:advanced/command_data
```

71.8 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).

71.8.1 Syntax

```
imsimta cnbuild
```

Table 71.8 imsimta cnbuild Command Switches

Switch	Default
`-image_file=`*`file-spec`*	`-image_file=IMTA_TABLE:advanced/config_data`
`-maximum`	`-nomaximum`
`-option_file=`*`file-spec`*	`-option_file=IMTA_TABLE:option.dat`
`-remove`	`-noremove`
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`
`-check`	`-nocheck`
`-synonyms`	`-nosynonyms`
`-xml_config`	`-noxml_config`

71.8.1.1 Restrictions

Must have superuser privileges, or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.8.2 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_TABLE: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

- *CONFIGROOT*`/confdesc.xml`,
- *CONFIGROOT*`/reverse.txt`,
- *CONFIGROOT*`/general.txt`, and
- *CONFIGROOT*`/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:

- *CONFIGROOT*`/imta.cnf`,
- *CONFIGROOT*`/aliases`,
- *CONFIGROOT*`/mappings`,
- *CONFIGROOT*`/option.dat`,
- *CONFIGROOT*`/conversions`,
- *CONFIGROOT*`/imta.filter`,
- *CONFIGROOT*`/reverse.txt`,
- *CONFIGROOT*`/general.txt`, and
- *CONFIGROOT*`/forward.txt`,

as well as the file *CONFIGROOT*`/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
```

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 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 interatively 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 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.

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.)

71.8.3 Switches

71.8.3.1 -check, -nocheck (default)

-check means to check the structure of the confdesc.xml file.

71.8.3.2 -image_file[=*file-spec*], -noimage_file

By default, imsimta cnbuild creates as output the image file named *CONFIGROOT*/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.

71.8.3.3 -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.

71.8.3.4 -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 *CONFIGROOT*/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, *CONFIGROOT*/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 *CONFIGROOT*/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.

71.8.3.5 -remove, -noremove (default)

The -remove switch causes removal of any existant compiled configuration; *i.e.*, removes the file *CONFIGROOT*/advanced/config_data (located prior to MS 7.0.5 via the imta_config_data MTA Tailor file option).

71.8.3.6 -sizes, -nosizes (default)

The -sizes switch instructs imsimta cnbuild to output information on the sizes of uncompiled MTA tables.

71.8.3.7 -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.

71.8.3.8 -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...
733 -> percents
822 -> sourceroute
brightmail -> sourcespamfilter1
channelfilter -> destinationfilter
...additional aliased channel keywords...
```

71.8.3.9 -xml_config, -noxml_config (default)

-xml_config directs cnbuild to compile a Unified Configuration; -noxml_config directs cnbuild to compile a legay configuration.

71.8.4 Examples

```
# 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.)

```
# imsimta cnbuild -noimage_file -option_file -maximum
# imsimta cnbuild
```

Use these two UNIX commands when you encounter the infamous "`No room in table`" MTA error message.

71.9 connkill utility

iref item="Utilities" subitem="connutil"/>

Terminate MTA connections coming from a specified IP address or authenticated using a specified account. The termination can be a one-time thing only affecting current connections matching the given criteria (the default), or it can be "sticky" and made to apply to future connections until a timeout value is reached. The termination request can also be applied to all dispatcher services (the default), or only to services whose names match a specified pattern.

Note: At present only SMTP services respond to the kill requests sent by this utility.

71.9.1 Syntax

```
imsimta connkill -ip=<ip-address>
imsimta connkill -user=<user>
```

Table 71.9 imsimta connkill Command Switches

Switch	Default
`-ip=`*`ip-address`*	n/a
`-service=`*`service-pattern`*	`-service=*`
`-sticky`	`-nosticky`
`-user=`*`uid@domain`*	n/a

71.9.1.1 Restrictions

Must have superuser privileges, or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.9.2 Description

Compromised hosts and user accounts are a fact of life in modern email environments. Blocking connections from specific IP addresses and suspending user accounts, respectively, are the usual of dealing with these threats.

However, blocking future connections doesn't affect currently active connections, which can continue to send mail until transaction limits are reached, connections time out, and so on.

It is therefore desirable to have some means of terminating active connections based on IP address or authentication state. The `imsimta connkill` utility provides this capability.

The command has two basic forms:

- The `-ip` switch is used to specify an IP address. In this case connections originating from this IP address are terminated.
- The `-user` switch is used to specify a user name of the form *uid@domain*. In this case connections that have authenticated to the account with the specified *uid* and in the specified domain are terminated.

At least one of `-ip` and `-user` are required, but they cannot be specified simultaneously. Both switches require an argument.

In both cases termination will occur the next time a command is read from the client.

Normally only current connections are affected - a subsequent connection from the specified IP address or authentication using the specified user will be allowed. The optional `-sticky` switch can be used to change this behavior. If `-sticky` is specified, subsequent connections or authentications will result in an immediate disconnect until the timeout in seconds specified by the `KILLED_IP_TIMEOUT` and `KILLED_USER_TIMEOUT` TCP/IP channel-specific options, respectively. The former defaults to twice the value of the `COMMAND_RECEIVE_TIME` TCP/IP channel-specific option; the latter defaults to 600 seconds.

Kill requests are sent to all dispatcher services by default. The `-service` switch can be used to specify which services requests are sent to. The required argument is a pattern the service name must match in order to receive the request.

71.10 counters -clear utility

Clear the in-memory cache of channel counters.

71.10.1 Syntax

```
imsimta counters -clear
```

71.10.1.1 Restrictions

The process must have the same UID as either the `root` or the MTA user (`user` option in `restricted.cnf`) accounts.

71.10.2 Parameters

None.

71.10.3 Description

To zero the in-memory channel counters, issue an `imsimta counters -clear` command.

The `imsimta counters -clear` command will create a new memory section, if one does not already exist. The values in the in-memory section will be set to zero, and then the stored messages, recipients, and volumes fields will be set from the values currently in the MTA queue cache database.

Since some initial values will be set based on entries in the MTA queue cache database, you may wish to issue the UNIX command

```
# imsimta cache -synchronize
```

before clearing the counters, to ensure that the MTA queue cache database entries are current.

71.11 counters -show utility

Display the contents of the in-memory cache of channel counters.

71.11.1 Syntax

```
imsimta counters -show
```

Table 71.10 imsimta counters -show Command Switches

Switch	Default
`-associations`	`-associations`
`-channels[=`*`channel-name`*`]`	`-channels`
`-debug`	`-nodebug`
`-format=`*`n`*	`-format=0`
`-headers`	`-headers`
`-output=`*`file-spec`*	*None*

71.11.1.1 Restrictions

On UNIX, normally none, but if a new in-memory section must be created then privileges sufficient to create such a section are required.

71.11.2 Description

The contents of the in-memory cache of channel counters may be displayed with the `imsimta counters -show` command.

A new in-memory section will be created if one does not already exist. Note that if a new in-memory section must be created, the initial values for the number of messages stored, number of recipients, and message volumes will be set based on the entries in the MTA queue cache database.

See MTA counters, especially the discussion of the Purpose and design of MTA counters.

71.11.3 Switches

71.11.3.1 `-associations` (default), `-noassociations`

The `-associations` switch specifies whether to show the in-memory cache of association counters.

71.11.3.2 `-channels[=`*`channel-name`*`]` (default), `-nochannels`

The `-channels` switch specifies whether to show the in-memory cache of channel counters. By default, counters for all channels are shown; specifying an argument to the `-channels` specifies a specific channel for which to show counters.

71.11.3.3 -debug, -nodebug (default)

The -debug qualifer enables debugging.

71.11.3.4 -format=0 (default=0)

Controls the format of the command output. The following values are supported:

0 Human-readable tabuular text

10 CVS format

11 JSON format, zero values omitted

12 JSON format, no values omitted

71.11.3.5 -headers (default), -noheaders

Controls whether or not a header line describing each column in the table of counters is output.

71.11.3.6 -output=*file-spec*

Direct the output to the specified file. By default the output appears on your display.

71.11.4 Examples

This example shows displaying the counters for all channels and associations.

```
# imsimta counters -show
Channel                     Messages  Recipients      Blocks
-----------------------  ----------  ----------  ----------
ims-ms
    Received                    3863        3881       25786
    Stored                        89          89         460
    Delivered                   3876        3894       26018 (3859 first time)
    Submitted                      1           1           7
    Attempted                     17          17          25
    Rejected                       0           0           0
    Failed                         1           1           6

    Queue time/count        29794837/3877 = 7.68502E3
    Queue first time/count  18904343/3860 = 4.8975E3

tcp_local
    Received                     208         217        4153
    Stored                         3           3           9
    Delivered                    200         212        2461 (197 first time)
    Submitted                   4053        4078       25919
    Attempted                      7           7           0
    Rejected                      46          68           0
    Failed                        14          14        1695

    Queue time/count        1106266/211 = 5.24297E3
    Queue first time/count  455897/208 = 2.19181E3
```

```
    Current In Assocs            127
    Total In Assocs             1056
    Total Out Assocs             132
    Rejected Out Assocs           11
    Failed Out Assocs              1

Channel      Timestamp    Association
------------ ------------ -------------------------------------------------
tcp_local    01-
Feb 00:27 TCP|192.160.253.70|25|192.160.253.66|3465
tcp_local    25-Jan 00:31 TCP|192.160.253.70|25|192.160.253.66|3496
tcp_local    26-Jan 14:50 TCP|192.160.253.70|25|192.160.253.66|2086
tcp_local    05-Feb 12:23 TCP|192.160.253.70|25|192.160.253.66|3593
tcp_local    01-Feb 00:34 TCP|192.160.253.70|25|192.160.253.66|3581

...
#
```

71.12 crdb utility

`imsimta crdb` is a utility used to create and update MTA (on-disk) database files.

71.12.1 Syntax

```
imsimta crdb input-file-spec output-database-spec
```

Table 71.11 imsimta crdb Command Switches

Switch	Default
`-append`	`-noappend`
`-count`	`-count`
`-dump`	*See text*
`-duplicates`	`-noduplicates`
`-exception_file=`*`file-spec`*	`-noexception_file`
`-huge_records`	`-huge_records`
`-long_records`	`-nolong_records`
`-quoted`	`-noquoted`
`-remove`	`-noremove`
`-statistics`	`-statistics`
`-strip_colons`	`-nostrip_colons`

71.12.2 Parameters

71.12.2.1 *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.

71.12.2.2 *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`.

71.12.3 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.

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.

71.12.4 Switches

71.12.4.1 `-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.

71.12.4.2 `-count` (default), `-nocount`

Controls whether or not a count is output after each group of 100 input lines are processed.

71.12.4.3 `-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.

71.12.4.4 `-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.

71.12.4.5 `-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.

71.12.4.6 `-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.

71.12.4.7 `-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.

71.12.4.8 `-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.

71.12.4.9 `-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.

71.12.4.10 `-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.

71.12.5 Examples

```
# imsimta crdb -long_records IMTA_DATAROOT:db/aliases.txt IMTA_DATAROOT:db/tmpdb
# imsimta renamedb IMTA_DATAROOT:db/tmpdb IMTA_DATAROOT: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.

71.13 find utility

Find specified version of an MTA log file.

71.13.1 Syntax

```
imsimta find file-pattern
```

Table 71.12 imsimta find Command Switches

Switch	Default
`-debug`	*None*
`-f=offset-from-first`	*None*
`-first=offset-from-first`	*None*
`-l=offset-from-last`	*None*
`-last=offset-from-last`	*None*

71.13.1.1 Restrictions

Must have read access to the requested file.

71.13.2 Parameters

71.13.2.1 *file-pattern*

A file name pattern for which MTA debug log file to find. If no directory is specified, the utility looks in the MTA log directory (`imta_log`). Note that the utility does not support relative directory specifications; when specifying a directory, use an absolute directory specification.

71.13.3 Description

The `imsimta find` utility may be used to find the precise file name of the specified "version" of an MTA debug log file. MTA debug log files have a *-uniqueid* appended to the file name to allow for the creation of multiple "versions" of the log file; on UNIX, the *-uniqueid* is appended to the very end of the file name (the end of the file extension). (See for instance the `master_debug` and `slave_debug` channel options, and the `dispatcher.debug`job_controller.debug options.) The `imsimta find` utility understands these unique ids and can find the particular file name corresponding to the requested "version" of the file.

The default, if no offset switch is specified, is to find the most recent "version" of the file.

71.13.4 Switches

71.13.4.1 `-debug`

The `-debug` switch enables debug output.

71.13.4.2 -f=*offset-from-first*, -first=*offset-from-first*

The -f or -first switch is used to specify finding the *n*th "version" of the file (starting counting from 0). For instance, to find the earliest (oldest) "version" of the file, specify -f=0

71.13.4.3 -l=*offset-from-last*, -last=*offset-from-last*

The -l or -last switch is used to specify finding the *n*th from the last "version" of the file (starting decrementing from 0 as the most recent version). For instance, to find the most recent (newest) "version" of the file, specify -l=0

71.13.5 Examples

```
# imsimta find tcp_local_slave.log
```

The above UNIX command will print out the file name of the tcp_local_slave.log-*uniqueid* file most recently created in the MTA log directory.

```
# imsimta find tcp_local_master.log -f=0
```

The above UNIX command will display the file name of the oldest tcp_local_master.log-*uniqueid* file in the MTA log directory.

71.14 process utility

List currently executing MTA processes and jobs.

71.14.1 Syntax

```
imsimta process
```

71.14.2 Parameters

None.

71.14.3 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.

71.14.4 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
```

71.15 purge utility

Purge MTA log files.

71.15.1 Syntax

```
imsimta purge [file-pattern]
```

Table 71.13 imsimta purge Command Switches

Switch	Operation	Default
`-day=value`	Days after which the file should be deleted.	None
`-debug`	Display the action(s) purge would take but do not perform them	None
`-hour=value`	Hours after which the file should be deleted.	None
`-num=value`	`-num=5`	

71.15.2 Parameters

71.15.2.1 *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. (The MTA log directory has symbolic name IMTA_LOG:, which is the directory *DATAROOT*`/log`. 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.

IMPORTANT NOTE: imsimta knows nothing about the specifics of the log files it operates on; it uses generic file name patterns which can effect ANY file in the log directory. As such, administrators MUST avoid creating their own files in the log directory - whether that be a rename of logs the product creates, any derivative material from the logs (parsing output, tar, zip, ... any), or anything completely unrelated - putting it in the log directory is unsupported and may produce unexpected results.

If you want to do your own parsing and archiving of the logs,then that is fine, but make sure you put the results into some other directory.

As of 8.0.1.2, imsimta purge unconditionally ignores the connection and transaction log files mail.log, mail.log_current, and mail.log_yesderday and connection log files connection.log, connection.log_current, and connection.log_yesterday. Files with names beginning with "." are also ignored.

71.15.3 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.)

71.15.4 Switches

71.15.4.1 -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.

71.15.4.2 -debug

Specifying -debug enables some debug output to stdout.

71.15.4.3 -hour=*h*

Specifying -hour=*h* results in purging all but the last *h* hours worth of log files.

71.15.4.4 -num=*n*

Specifying -num=*n* results in purging all but the last *n* log files. The default is -num=5.

71.15.5 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/.

71.16 qclean utility

Hold or delete message files from the MTA queue disk area that contain specified substrings in their envelope From address, Subject: header, or message content.

71.16.1 Syntax

```
imsimta qclean [channel]
```

Table 71.14 imsimta qclean Command Switches

Switch	Default
`-content=`*`substring`*	*None*
`-database`	`-database`
`-delete`	`-hold`
`-directory_tree`	`-database`
`-domain_to=`*`substring`*	*None*
`-env_from=`*`substring`*	*None*
`-env_to=`*`substring`*	*None*
`-from=`*`substring`*	*None*
`-hold`	`-hold`
`-ignore_zz`	`-noignore_zz`
`-match=`*`keyword`*	`-match=AND`
`-min_length=`*`n`*	`-min_length=24`
`-subject=`*`substring`*	*None*
`-threads=`*`n`*	`-nothreads`
`-to=`*`substring`*	*None*
`-verbose`	`-noverbose`

71.16.1.1 Restrictions

Privileges sufficient to read and delete files in the MTA channel queue directory tree, as well as read the MTA queue cache database (obtain information from the Job Controller), are required.

71.16.2 Parameters

71.16.2.1 *channel*

Optional parameter which specifies a specific MTA channel area to be searched for matching messages. * or ? wildcard characters may be used in the channel specification.

71.16.3 Description

Hold or delete message files containing specific substrings in their envelope From address, Subject: line, or content. (Note that `.HELD` message files are not checked or affected by

this utility.) By default, message files are held (`-hold`). Specify `-delete` to instead delete matching message files. The `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` switches are used to specify the substrings for which to search. Often, the specific user recipient is not as relevant as the domain of the recipient(s), so the `-domain_to` switch is available for the purpose of specifying the recipient domain.

Any combination of `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` may be specified. However, only one of each may be used. The `-match` switch controls whether a message file must contain all (`-match=AND`, the default) or only one of (`-match=OR`) the specified substrings in order to be held or deleted. The default is `-match=AND`.

By default, each substring to be searched for must be at least 24 bytes long (`-min_length=24`). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the `-min_length` switch to override this limit, making sure to specify `-min_length` *before* any switches specifying "short" strings. And note that only values for `-min_length` of 5 or more will be respected; attempting to set a `-min_length` of less than 5 will result in a value of 5 being used. Also by default, only message files identified by the queue cache database are searched (`-database`). Use the `-directory_tree` switch to instead search all message files actually present in the channel queue directory tree.

The optional channel parameter restricts the search to message files in the specified channel. The channel parameter may use * and ? wild cards.

The `-threads` switch may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be in the range 1-8. The default is `-nothreads`.

Note that this utility does not bother to update the Job Controller's queue cache regarding the `.HELD`ing or deleting of message files; the utility merely makes the change directly to the message file on disk. (The Job Controller, when it happens to attempt to access and process such a modified or deleted message file, will see that the message file no longer exists in its previous form and then update its own data structure and continue on to process other messages.)

71.16.4 Switches

71.16.4.1 `-content=`*`substring`*, `-domain_to=`*`substring`*, `-env_from=`*`substring`*, `-env_to=`*`substring`*, `-subject=`*`substring`*, `-to=`*`substring`*

The `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` switches are used to specify the substrings for which to search. Any combination of `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` may be specified; explicit use of at least one such switch is required. However, only one occurrence of each such switch may be used. When a combination of such switches is used, the `-match` switch controls whether the switches are interpreted as further restrictions (`-match=AND`), or as alternatives (`-match=OR`).

71.16.4.2 `-database` (default), `-directory_tree`

The `-database` switch, the default, specifies that only message files identified by the Job Controller's queue cache be searched. Use the `-directory_tree` switch to instead search all message files actually present in the channel queue directory tree.

71.16.4.3 `-delete`, `-hold` (default)

`-hold` is the default and means that matching message files will be sidelined as `.HELD` files. Specify `-delete` to instead delete matching message files.

71.16.4.4 `-match=`*keyword*

The default is `-match=AND`, meaning that any criteria specified by `-content`, `-env_from`, and `-subject` switches must all match in order for the current hold or delete operation to be applied. Specifying `-match=OR` means that a message will match as long as at least one such criterion matches.

71.16.4.5 `-min_length=`*n*

By default, each substring to be searched for must be at least 24 bytes long (`-min_length=24`). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the `-min_length` switch to override this limit, making sure to specify `-min_length` *before* any switches specifying "short" strings.

Note that the `-min_length` value must be 5 or more; attempting to set a smaller value will result in the value 5 being used.

71.16.4.6 `-threads=`*n*, `-nothreads` (default)

The `-threads` switch may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be an integer in the range 1-8. The default is `-nothreads`.

71.16.4.7 `-verbose`, `-noverbose` (default)

The `-verbose` switch may be used to request that the utility print out information about what it is doing as it operates.

71.16.5 Examples

```
# imsimta qclean -min_length=12 -subject="make money fast" -env_from="spammers.com"
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds (19.20 messages/second )
%QM-I-HELD, held 5 message files
```

The above UNIX example shows holding all message files in the MTA queue area that have the string "make money fast" in the Subject: header and have the string "spammers.com" in the envelope From address.

71.17 qtop utility

Display the most frequently occurring envelope From, envelope To, Subject:, or message content fields found in message files in the channel queues.

71.17.1 Syntax

```
imsimta qtop [channel]
```

Table 71.15 imsimta qtop Command Switches

Switch	Default
-content=*offset-specifier*	*None*
-database	-database
-directory_tree	-database
-domain_to=*offset-specifier*	*None*
-env_from=*offset-specifier*	*None*
-env_to=*offset-specifier*	*None*
-from=*offset-specifier*	*None*
-ignore_zz	-noignore_zz
-min_count=*n*	-min_count=2
-output=*file-spec*	*See text*
-subject=*offset-specifier*	-subject=(START=1,LENGTH=2147483647)
-threads=*n*	-nothreads
-to=*offset-specifier*	*None*
-top=*n*	-top=20
-verbose	-noverbose

71.17.1.1 Restrictions

Privileges sufficient to read files in the MTA channel queue directory tree, as well as read the MTA queue cache database, are required.

71.17.2 Parameters

71.17.2.1 *channel*

Optional parameter which specifies a specific MTA channel area to be scanned for string frequencies. * or ? wildcard characters may be used in the channel specification.

71.17.3 Description

Display the most frequently occurring envelope From, envelope To, Subject: field, or message content fields found in message files in the channel queues. (Note that .HELD

message files are not checked or displayed by this utility.) By default, only Subject: fields are shown (`-subject`). Use `-env_from` to display frequent envelope From fields, `-env_to` or `-domain_to` to display frequent envelope To fields, or `-content` to display frequent message contents. Any combination of `-content`, `-env_from`, `-env_to`, and `-subject` may be specified. However, only one of each may be used.

The optional channel parameter restricts the scan to message files in the specified channel. The channel parameter may use * and ? wild cards.

By default, the top 20 most frequently occurring fields are shown (`-top=20`) provided that they occur 2 or more times (`-min_count=2`). Use the `-top` and `-min_count` switches to alter this behavior. Also by default, only message files identified by the queue cache database are scanned (`-database`). Use the `-directory_tree` switch to instead scan all message files actually present in the channel queue directory tree.

The `-threads` switch may be used to accelerate scanning on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous scanning threads, specify `-threads=n`. The value *n* must be in the range 1-8. The default is `-nothreads`.

The `-content`, `-env_from`, `-env_to`, and `-subject` switches accept the optional qualifiers `start=`*n* and `length=`*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are `-content=(START=1,LENGTH=256)`, `-env_from=(START=1,LENGTH=2147483647)`, `-env_to=(START=1,LENGTH=2147483647)`, and `-subject=(START=1,LENGTH=2147483647)`. Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

71.17.4 Switches

71.17.4.1 -content[=*offset-specifier*], -domain_to[=*offset-specifier*], -env_from[=*offset-specifier*], -env_to[=*offset-specifier*], -from[=*offset-specifier*], -subject[=*offset-specifier*], -to[=*offset-specifier*]

The `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` switches are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Use `-env_from` to display frequent envelope From fields, `-env_to` or `-domain_to` to display frequent envelope To fields, or `-content` to display frequent message contents. Any combination of `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` may be specified. However, only one of each may be used. The `-content`, `-domain_to`, `-env_from` (synonym `-from`), `-env_to` (synonym `-to`), and `-subject` switches accept the optional qualifiers `START=`*n* and `LENGTH=`*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are `-content=(START=1,LENGTH=256)`, `-env_from=(START=1,LENGTH=2147483647)`, `-env_to=(START=1,LENGTH=2147483647)`, and `-subject=(START=1,LENGTH=2147483647)`. Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

71.17.4.2 -database (default), -directory_tree

The -database switch, the default, specifies that only message files identified by the queue cache database be searched. Use the -directory_tree switch to instead search all message files actually present in the channel queue directory tree.

71.17.4.3 -ignore_zz, -noignore_zz (default)

-noignore_zz is the default and means to scan all messages queued in a channel. -ignore_zz means to scan only those files which are not ZZ files; that is, only those files which have already had at least one delivery attempt.

71.17.4.4 -min_count=*n*

By default, a string must occur at least 2 times, -min_count=2, in order to be displayed.

71.17.4.5 -output=*file-spec*

By default, output goes to stdout. The -output switch may be used to direct the utility's output to a file instead.

71.17.4.6 -threads=*n*, -nothreads (default)

The -threads switch may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer in the range 1-8. The default is -nothreads.

71.17.4.7 -top=*n*

By default, the top 20 most frequently occurring fields are shown, (-top=20).

71.17.4.8 -verbose, -noverbose (default)

The -verbose switch may be used to request that the utility print out information about what it is doing as it operates.

71.17.5 Examples

```
# imsimta qtop -subject -env_from
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Envelope From: addresses which occur 2 or more times
 Count  Envelope From: address
======  ============================================================
    27
    10  owner-ex-list@acme.com
     2  owner-test-list@acme.com

Top 20 Subject: header lines which occur 2 or more times
```

```
 Count  Subject
======  ========================================================================
     6  Re: your ex-list posting
     2  Test posting to test-list
```

The above UNIX example shows displaying the most frequently occurring Subject: field and envelope From addresses amongst messages in the MTA queue area.

```
# imsimta qtop -subject=START=12 -min_count=15
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Subject: header lines which occur 15 or more times
 Count  Subject
======  ========================================================================
    25  ake money fast $$$
```

The above UNIX example shows displaying the most frequently occuring Subject: lines that occur 20 times or more, starting from 12 characters into the Subject: header value. This may be useful when trying to spot spam that inserts random characters at the beginning of the Subject: header value.

71.18 reload utility

Reload portions of the MTA configuration.

71.18.1 Syntax

```
imsimta reload
```

71.18.2 Parameters

None.

71.18.3 Description

For compiled configurations, this utility reloads any mapping tables (`mapping` group settings in Unified Configuration or the mapping file in legacy configuration), and the general, reverse, and forward "databases", if the applicable bits of `use_text_databases` have been set. It does not, however, reload the entire compiled configuration; in particular, changes to channel definitions and rewrite rules (*e.g.*, the `imta.cnf` file in legacy configuration) are not propagated by `imsimta reload`. Nor does it, in general, reload changes to MTA options.

The `imsimta reload` utility also reloads authentication, SSL/TLS, and as of Messaging Server 8.0.2, affinity group options. This is done even if the MTA's configuration is not compiled.

For making certain changes "live" to a running Job Controller, see also the `cache -change` utility.

71.18.4 Error messages

Problems encountered attempting to reload the configuration can result in a syslog notice (sent with syslog facility and mask controlled by the `sndopr_priority` MTA option):

```
BADCONFIGLOAD, Attempt to load new configuration failed.
```

or

```
BADCONFIG, Attempt to map new configuration failed.
```

71.19 restart utility

Restart MTA components.

71.19.1 Syntax

```
imsimta restart [component]
```

71.19.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility.

71.19.2 Parameters

71.19.2.1 *component*

Optional parameter which specifies a specific MTA component, or Dispatcher service, to be restarted. The components which may be restarted with this utility are `job_controller`, `dispatcher`. Any Dispatcher service may be restarted with this utility by specifying the service name; with the services in a typical configuration, this means `smtp`, `smtp_submit`, and `lmtpss`, but any other site-specific services may be restarted similarly by specifying their name. Note that restarting the MTA Dispatcher, *i.e.*, the `dispatcher` component, effectively restarts all the service components it handles, which may include `smtp`, `smtp_submit`, and `lmtpss`. If no component name is given, then all active components will be restarted. If the asterisk character, `*`, is specified as the "component", then all active services under the Dispatcher (but not the Dispatcher itself) will be restarted; note that to avoid shell interpretation and get the asterisk character passed through as an argument to the utility itself, it will usually be necessary to quote the asterisk character, `"*"`.

71.19.3 Description

Detached MTA processes should be restarted whenever relevant portions of the MTA configuration are altered---these processes load information from the configuration once only and generally need to be restarted in order for configuration changes to become visible to them. (In legacy configuration, in addition to general MTA configuration files such as the `imta.cnf` file, note that components such as the Dispatcher have their own specific configuration files, *e.g.*, `dispatcher.cnf`, and should be restarted after changes to any of these files.) However, note that *some* configuration changes can be put into effect without a restart: see the `imsimta reload` and `imsimta cache -change` utilities.

An `imsimta restart` command is equivalent to performing a `imsimta shutdown` followed by an `imsimta startup` on the specified component.

If no component name is specified, then the `imsimta restart` utility stops any old Job Controller process and Dispatcher process that might be running, and restarts the Job Controller and Dispatcher. If a component parameter is specified, then only detached processes associated with that component will be restarted. If an asterisk character, `*`, is specified, then all active services running under the Dispatcher will be restarted (though the Dispatcher itself will not be restarted in this case); note that shell interpretation will usually

require that the asterisk character be quoted so that the shell will pass the asterisk through as an argument to the `imsimta restart` command.

The standard component names are:

Table 71.16 Component Names

Component	Description
`dispatcher`	MTA multithreaded Dispatcher handling services such as SMTP, SMTP SUBMIT, and LMTP servers.
`job_controller`	MTA Job Controller.
`lmtpss`	LMTP server for Message Store delivery
`smtp`	SMTP server processes.
`smtp_submit`	SMTP SUBMIT (port 587) server processes.

Note that restarts of the Job Controller should be avoided, especially during periods of large message backlogs, unless such a restart is necessary. Note that a restart of the Job Controller requires that the Job Controller rescan the queue directory structure to re-discover all message files, and completely rebuild its internal data structures that control message delivery scheduling and the order of message delivery attempts. Such rescanning and rebuilding of data structures is designed to be fast, and have a low impact on the processing of any newly submitted messages that may come in during the rebuilding. But it is not intended to necessarily restore the exact state of the Job Controller's original scheduling, and in practice a certain degree of "out-of-order" processing of the backlogged messages (messages already existing on disk), and "unfair" division between channels of effort on processing the backlogged messages, is expected to occur after a restart of the Job Controller.

Whenever the Job Controller is shutdown (and a restart includes a shutdown of the "old" Job Controller process), before it shuts down it sends a "no more messages" notice to all its child processes (channel processes), which such processes will see once they finish up whatever current message they are working on and ask the Job Controller whether there are any more messages. That is, the Job Controller itself shuts down, but leaves a "poison" message that the channel jobs will see as they each finish up their own work. So generally (but with a couple of exceptions, to be discussed just below) any old channel processes that the Job Controller had previously started finish up whatever message they had already been working on. Once a channel process has finished its already started work and then asks for another message to process, it will see that there are no more messages for it to process, and exit. However, exceptions are the `ims-ms` channel, and less commonly, outbound TCP/IP channels. `ims-ms` channels will interrupt processing even of currently being processed messages after a brief amount of time (an additional 55 seconds) to attempt to complete processing of a current message; the remaining recipients of a message for which processing had begun will be deferred for later processing; (when logging is enabled, "Z" records with a reason, as of MS 8.0, of "`shutting down`" will be generated for those recipients; note that prior to MS 8.0, the reason was was "`shutdown`"). TCP/IP channels will abort processing after their configurable `WINDDOWN_TIMEOUT` channel-specific option time period has expired. (The default value for `WINDDOWN_TIMEOUT` is relatively "large" due to the potential for undesirable remote SMTP server handling of aborted message transfers, and the potential expense of resending a "large" message.)

71.19.4 Examples

```
# imsimta restart dispatcher
```

The above UNIX command restarts the Dispatcher, and hence all services under the dispatcher.

```
# imsimta restart "*"
```

The above UNIX command restarts all the services *under* the Dispatcher (typically `SMTP`, `SMTP_SUBMIT`, and `LMTPSS`) *without* restarting the Dispatcher itself.

```
# imsimta restart
```

The above UNIX command restarts *all* the MTA jobs, including the Job Controller (and should be avoided unless a restart of the Job Controller is truly needed).

```
# imsimta restart
Stopping dispatcher server 29615 .. done
Stopping job_controller server 29637 . done
Starting dispatcher server . 29645
Starting job_controller server . 29655
```

The above shows the sort of output shown as of MS 6.3, including in particular the pids of the stopped and started processes.

```
# imsimta restart job_controller
job_controller server is not running
Starting job_controller server .. 29683
```

The above shows the sort of output shown as of MS 6.3, if a `restart` command is given for the Job Controller when it is not actually currently running.

71.19.5 Error messages

Trouble communicating with the Dispatcher, shutting it down, or restarting it, will be reported back to `stderr`:

```
dispatcher server is not running
```

```
Cannot stop dispatcher server (pid=pid) with SIGTERM
Retrying with SIGKILL
```

```
dispatcher server is running already
```

Trouble restarting the Job Controller will be reported back to `stderr`:

```
job_controller server is not running
```

```
job_controller server is running already
```

71.20 return utility

Return (bounce) a mail message to its originator.

71.20.1 Syntax

```
imsimta return message-file-spec
```

71.20.1.1 Restrictions

Must have superuser privileges or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.20.2 Parameters

71.20.2.1 *message-file-spec*

File specification of the message file to return. The specification may include wildcards, but if so, the specification must be quoted.

71.20.3 Description

The `imsimta return` utility returns a message to the message's originator, (unless the original message had NOTIFY=NEVER set, in which case the message is simply deleted without generation of a DSN). The returned message is in three parts. The first part explains the reason why the message is being returned; the text of the reason is contained in the file `return_bounced.txt` file located in the MTA language-specific directory. The second part of the returned message is a machine-readable report. The third part contains either the original message itself, or the original message's header (depending upon any RET=FULL *vs.* RET=HDRS setting on the original message, and on whether the original message size triggered application of MTA size limits such as `content_return_block_limit`).

As regards MTA transaction logging, when it is enabled via the `logging` channel option, such postmaster manual bounces will result in either a "R" (bounce original message with a DSN) or "K" (delete original message without a DSN since the original message requested NOTIFY=NEVER) MTA transaction log entry.

71.20.4 Examples

```
# imsimta return 'IMTA_QUEUE:ims-ms/*'
```

The above UNIX command will cause all of the messages currently in the `ims-ms` channel queue to be returned to their respective originators (*i.e.*, "bounced").

71.21 run utility

Process messages in a specified channel.

71.21.1 Syntax

```
imsimta run channel [poll-flag [start-time [start-id
  [channel-specific-argument [name-filter [min-priority:max-priority]]]]]]
```

71.21.1.1 Restrictions

Must be the MTA user (see the `user` option in `restricted.cnf`), or have superuser privileges (in which case the MTA will automatically become the MTA user) in order to use this utility.

71.21.2 Parameters

71.21.2.1 *channel*

This parameter specifies the channel to be processed.

71.21.2.2 *poll-flag*

Valid values for *poll-flag* are `nopoll` or its synonym `master` (the default), `poll`, or `slave`. By default, if neither `poll` nor `slave` is specified, then only the master direction of the channel will be run. If `poll` is specified, then first the master and then the slave directions of the channel will be run. If `slave` is specified, then only the slave direction of the channel will be run.

71.21.2.3 *queuename*

This parameter is of historical interest only, and has no effect when the Job Controller is in use.

71.21.2.4 *start-time*

This parameter is of historical interest only, and has no effect when the Job Controller is in use.

71.21.2.5 *start-id*

This parameter is of historical interest only, and has no effect when the Job Controller is in use.

71.21.2.6 *channel-specific-argument*

This parameter has no effect.

71.21.2.7 *name-filter*

For TCP/IP channel master jobs and for `ims-ms` channel jobs, the *name-filter* may be used to specify a message file name filter and/or a (destination) host name filter, using the syntax

file-name-filter@host-name-filter

This parameter is not respected by other channels.

71.21.2.8 *min-priority:max-priority*

This parameter is of historical interest only, and has no effect when the Job Controller is in use.

71.21.3 Description

The `imsimta run` utility processes the messages in the channel specified by the *channel* parameter.

In order to run the channel "outside" normal Job Controller processing (and the Job Controller's normal limits on job creation---that being one of the reasons that `imsimta run` is sometimes used, to force execution of a channel that the Job Controller would not normally want to run at that moment), a sort of light-weight, pseudo-Job Controller runs in your process, and it then forks the actual channel job. So some Job Controller-like output (Job Controller error output, or debug output if you have Job Controller debugging enabled) may be output at your terminal, and your terminal will be "tied up" for the duration of the operation of the utility, though the actual channel execution will be forked to a child process (which if channel debugging is enabled, will write its own debug log file, as usual).

See also the `imsimta submit` utility (or synonymous `imsimta submit_master` utility), which, unlike `imsimta run`, submits a request to start a job to the Job Controller (and the Job Controller then forks the actual channel job, as appropriate), so these other commands will not tie up your terminal.

As alluded to above, one of the typical purposes of using `imsimta run` is to start up an "extra" channel processing job, exempt from both the Job Controller's usual rules for triggering creation of a new job, and its configured limits on creation of channel processing jobs. As of MS 6.3, the `imsimta submit` utility will also trigger a new job creation (be exempt from the usual rules for triggering creation of a channel job), though only within the Job Controller's configured limit on the maximum number of channel processing jobs allowed. That is, either `imsimta run` or `imsimta submit` may be used to attempt to trigger an "additional" channel job---but only `imsimta run` guarantees that the request will in fact result in another job (possibly an "extra" job beyond the Job Controller's configured limits on numbers of jobs). Triggering creation of another job might be desired, for instance, when attempting to process a "large" backlog of messages; and creation of an "extra" job (beyond the usual `maxjobs` and `job_limit` limits) via `imsimta run` might be desired when you know that you want "extra" jobs beyond the Job Controller's configured limits.

Note that since `imsimta run` is using special routines to simulate Job Controller channel job creation from within your own process, it is not typically useful for purposes of viewing/debugging actual Job Controller operation.

71.21.4 Examples

```
# imsimta run tcp_local poll
```

The above UNIX command may be used to process any messages in the `tcp_local` channel.

71.22 shutdown utility

Shut down MTA components.

71.22.1 Syntax

```
imsimta shutdown [component]
```

71.22.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility.

71.22.2 Parameters

71.22.2.1 *component*

The optional *component* parameter specifies a specific MTA component to be shut down. The components that may be specified are `job_controller`, `dispatcher`, and any services defined in the Dispatcher configuration, which typically might include `smtp`, `smtp_submit`, and `lmtpss`.

71.22.3 Description

The `imsimta shutdown` utility shuts down the MTA Job Controller and the MTA Dispatcher.

Note that shutting down the MTA Dispatcher causes a graceful shutdown of all services (*e.g.*, `SMTP`, `SMTP_SUBMIT`, `LMTPSS`) being managed by the MTA Dispatcher. That is, existing such server processes "finish up" whatever connections they currently have open, but do not accept new connections. As of MS 6.2p8, such server processes "finish up" processing of whatever message submissions were already in progress, but reject any attempts (even in the same, already opened connections) to submit further messages, with an effect as if `ALLOW_TRANSACTIONS_PER_SESSION`/`transactionlimit` had been reached. That is, prior to MS 6.2p8 while new connections would not be accepted, old connections might continue to work, accepting further message submissions; however, as of MS 6.2p8 even already existing connections are "poisoned" so that they may not receive additional new message submissions. (See the `disconnecttransactionlimit` channel option if you wish to have the server processes force a disconnect of connections once they will no longer accept further messages.)

Whenever the Job Controller is shutdown, before it shuts down it sends a "no more messages" notice to all its child processes (channel processes), which such processes will read from their communication buffer once they finish up whatever current message they are working on and wish to ask the Job Controller whether there are any more messages. That is, the Job Controller itself shuts down, but leaves a "poison" message in a buffer that the channel jobs will see as they each finish up their own work. So generally (but with a couple of exceptions, to be discussed just below) any old channel processes that the Job Controller had previously started finish up whatever message they had already been working on. Once a channel process has finished its already started work and then asks for another message to process, it will see that there are no more messages for it to process, and exit. However, exceptions are the `ims-ms` channel, and less commonly, outbound TCP/IP channels. When the Job Controller

has been shutdown, then `ims-ms` channels will interrupt processing even of currently being processed messages after a brief amount of time (an additional 55 seconds) spent attempting to complete processing of the current message; but the remaining recipients of a message for which processing had begun will be deferred for later processing; (when logging is enabled, "Z" records with a reason of "`shutdown`" will be generated for those recipients). TCP/IP channels will abort processing after their configurable `WINDDOWN_TIMEOUT` TCP/IP-channel-specific option time period has expired. (Note that the default value for `WINDDOWN_TIMEOUT` is relatively "large" due to the potential for undesirable remote SMTP server handling of aborted message transfers, and the potential expense of resending a "large" message.)

In typical Messaging Server configurations, `msprobe` may be configured to check periodically on the Job Controller, the SMTP server, and SMTP SUBMIT, and LMTP server; see the `msprobe.probe:job_controller.*`, `msprobe.probe:smtp.*`, `msprobe.probe:submit.*`, `msprobe.probe:lmtp.*`, and `schedule.task:msprobe.*` options in Unified Configuration. In particular, if with such configuration `msprobe` determines that the Job Controller, or a service managed by the Dispatcher, is not responding to its probe, and if autorestart has been enabled (`autorestart.enable` in Unified Configuration), then `msprobe` will ask the Watcher to restart the non-responsive process (such as the Job Controller or Dispatcher). So performing a `shutdown` of an MTA management process, or specific service (*e.g.*, `SMTP`) causes a temporary shutdown, but that shutdown is likely to be only temporary (unless such Messaging Server monitoring-and-automatic-restarting has not been enabled). (To shut down a service more permanently, the configuration should be more fundamentally modified: for instance, remove the Dispatcher's definition of the service in question, or to disable the Dispatcher or Job Controller, respectively, in Unified Configuration, set `dispatcher.enable` or `job_controller.enable` to `0`.) Thus the use/purpose of the `shutdown` command in modern usage tends to be to perform a *temporary* shutdown, perhaps while a configuration change is being made, with `imsimta startup` often used shortly thereafter, to start the component back up once a desired configuration change has been implemented.

71.22.4 Examples

```
# imsimta shutdown
Stopping dispatcher server 29020 ......... done
Stopping job_controller server 29637 . done
```

The above UNIX command shuts down the MS MTA, showing the sort of output shown in MS 6.3. (Earlier versions would not have as much/the same sort of output.)

```
# imsimta shutdown
dispatcher server is not running
job_controller server is not running
```

The above shows an example of the sort of output shown as of MS 6.3, if a `shutdown` command is issued when the MTA Dispatcher and Job Controller had not in fact been running.

71.22.5 Error messages

Trouble communicating with the Dispatcher, shutting it down, will be reported back to `stderr`:

```
dispatcher server is not running
```

or

```
Cannot stop dispatcher server (pid=pid) with SIGTERM
Retrying with SIGKILL
```

Trouble shutting down the Job Controller will be reported back to `stderr`:

```
job_controller server is not running
```

71.23 startup utility

Start the MTA Job Controller and the MTA Dispatcher.

71.23.1 Syntax

```
imsimta startup [component]
```

71.23.1.1 Restrictions

Must have superuser privileges or use the Solaris RBAC feature in order to use this utility.

71.23.2 Parameters

71.23.2.1 *component*

Optional parameter which specifies a specific MTA component to be started: `job_controller` or `dispatcher`. If no component name is given, then all active components will be started.

71.23.3 Description

The `imsimta startup` utility starts up detached MTA processes. If no component parameter is specified, then the MTA Job Controller and MTA Dispatcher are started. Note that starting the Dispatcher starts all services the Dispatcher is configured to handle, which may include, for instance, SMTP, SMTP SUBMIT, or LMTP servers. If a component parameter is specified, then only detached process associated with that component will be started. The standard component names are:

Table 71.17 `imsimta startup` Component Names

Component	Description
`dispatcher`	Dispatcher managing SMTP/SMTP SUBMIT/LMTP services
`job_controller`	Job Controller managing MTA channel processing jobs

Note that the services handled by the MTA Dispatcher must be started by starting (or restarting) the Dispatcher itself; only services *not* being handled by the Dispatcher can be individually started via the `imsimta startup` utility. (The Dispatcher may be configured to handle various services, *e.g.*, the SMTP, SMTP SUBMIT, and LMTP servers; see Dispatcher options for details.) Thus if adding a new service under the Dispatcher, or to reenable execution of a previously shutdown service which runs under the Dispatcher, on a running system you must restart the Dispatcher itself.

Since in Messaging Server usage, the overall Messaging Server startup command, `start-msg`, is normally used to start up not only the MTA but also other components of Messaging Server, the MTA-specific `imsimta startup` command is seldom used nowadays.

71.23.4 Examples

```
# imsimta startup
Starting dispatcher server ... 29034
Starting job_controller server . 29045
```

The above shows the sort of output shown as of MS 6.3, which includes the pids of the started Dispatcher and Job Controller processes.

```
# imsimta startup
dispatcher server is running already
job_controller server is running already
```

The above shows the sort of output issued in MS 6.3 if a `startup` command is issued when the MTA's Dispatcher and Job Controller were already running.

71.23.5 Error messages

Attempting to start up the Dispatcher when it is already running, or the Job Controller when it is already running, will be reported as an error to `stderr`:

```
dispatcher server is running already
```

```
job_controller server is running already
```

71.24 submit_master utility

Process messages in a specified channel.

71.24.1 Syntax

```
imsimta submit_master channel [poll-flag [host-pattern]]
```

Table 71.18 imsimta submit_master Command Switches

Switch	Default
`-delayed`	`-nodelayed`

71.24.1.1 Restrictions

Must have superuser privileges or be logged in as the MTA user (see the `user` option in `restricted.cnf`) in order to use this utility.

71.24.2 Parameters

71.24.2.1 *channel*

This required parameter specifies the channel to be processed.

71.24.2.2 *poll-flag*

Valid values for *poll-flag* are `nopoll` or its synonym `master` (the default), `poll`, or `slave`. By default, if neither `poll` nor `slave` is specified, the Job Controller will be requested to perform only a master direction execution of the channel (and the channel will not run unless there are actually messages waiting to be processed in that channel, whose `backoff` has expired). If `poll` is specified, the Job Controller will be requested to perform both master and slave direction executions of the channel. If `slave` is specified, the Job Controller will be requested to perform only a slave direction execution of the channel.

71.24.2.3 *host-pattern*

Specify a hostname for which to deliver. (This is not relevant for all channels, but for outgoing TCP/IP channels marked with the `single_sys` channel option, where the Job Controller maintains separate delivery queues for each distinct destination host, this parameter specifies which messages to consider delivering.)

71.24.3 Description

The `imsimta submit_master` utility requests that the Job Controller begin (consider beginning) a delivery job for the specified channel.

Note that stopped channels (`imsimta qm` utility's `stop` command) will not be run. And if there are already more than twice the `maxjobs` requests to run the channel, then this additional request will be ignored. A request will also be ignored if the backlog of messages

in the channel queue area is smaller than the number of already executing jobs for that channel times the `threaddepth` for the channel (unless there is only one executing job, in which case another job will be started). Note that the channel's `backoff` values will be respected in determing which, if any, messages to process. Thus it is rare that the `imsimta submit_master` utility is of much interest: generally, the Job Controller's intrinsic scheduling will already have started the maximum-appropriate-for-the-moment number of processing jobs for the channel, and so an `imsimta submit_master` command will have no immediate effect (other than causing the Job Controller to run through its checks before deciding that this additional request should be ignored). Since a limited (by `maxjobs`) number of such requests will be kept pending, in the medium term this utility may be used to attempt to influence the Job Controller to more aggressively schedule delivery of one destination host over another (within the same channel), via the *`host-pattern`* parameter; as the originally existing jobs terminate, the pending requests can begin to be honored.

See also the synonymous `imsimta submit` utility, and the `imsimta run` utility, which runs at your terminal outside the Job Controller's normally scheduling limits and checks (for instance, disregarding `backoff` conditions), rather than running as a regular Job Controller child process.

71.24.4 Switches

71.24.4.1 `-delayed`, `-nodelayed` (default)

By default (`-nodelayed`), the submitted job only considers and attempts processing of newly-submitted (have not yet received a delivery attempt) messages. To request delivery attempts for "old" messages (those that have already received one or more delivery attempts), use `-delayed`.

71.24.5 Examples

```
# imsimta submit_master tcp_local poll
```

The above UNIX command may be used to process any messages in the `tcp_local` channel.

71.25 submit utility

`imsimta submit` is an abbreviation for `imsimta submit_master`.

71.26 test -domain_map utility

Test a domain (domain map) lookup.

71.26.1 Syntax

```
imsimta test -domain_map [expression]
```

Table 71.19 imsimta test -domain_map Command Switches

Switch	Default
`-debug`	`-nodebug`
`-image_file`	`-noimage_file`
`-input=`*`filename`*	*None*
`-option_file=`*`file-spec`*	`-option_file=IMTA_TABLE:option.dat`
`-xml_config=`*`file-spec`*	*None*

71.26.2 Description

`imsimta test -domain_map` is an interactive utility for testing how domains are provisioned in LDAP.

Especially common uses/commands to use are to list (properly) provisioned domains in LDAP using the `ENUMERATE` command, or verify proper (as regards MTA requirements) provisioning of domains in LDAP using the `VERIFY` command.

Targetted probing of specific domain definitions can also be performed. A base command sequence is to `LOCATE` and then `SHOW` a domain:

```
DOMAIN_MAP> LOCATE DOMAIN mrochek.com
Entry located
DOMAIN_MAP> SHOW
Domain name: mrochek.com
Canonical name: mrochek.com
Lower case canonical name: mrochek.com
Base DN: o=mrochek.com,o=usergroup
Domain DN: dc=mrochek,dc=com,o=internet
```

After locating a domain, you can probe the value of a specific basic attributes:

```
DOMAIN_MAP> LOCATE DOMAIN mrochek.com
Entry located
DOMAIN_MAP> QUERY inetDomainBaseDN
Attribute value(s):
  [0] "o=doof,o-mrochek.org,o=usergroup"
DOMAIN_MAP> QUERY inetCanonicalDomainName
Attribute value(s):
  [0] "mrochek.org"
```

If you wish to know about other than the couple of basic attributes, then you have to let the utility know about such attributes via the ATTRIBUTE command *before* doing the LOCATE. *E.g.,*

```
DOMAIN_MAP> LOCATE DOMAIN mrochek.com
Entry located
DOMAIN_MAP> QUERY creatorsname
Not on known attribute list
DOMAIN_MAP> ATTRIBUTE creatorsname
Attribute now known to map
DOMAIN_MAP> LOCATE DOMAIN mrochek.com
Entry located
DOMAIN_MAP> QUERY creatorsname
Attribute value(s):
  [0] "cn=directory manager"
```

71.26.3 Commands

imsimta test -domain_map commands shows the interactive commands available.

Table 71.20 imsimta test -domain_map commands

Command	Purpose
ATTRIBUTE *attr-name*	Tell the utility the name of an LDAP attribute which you wish to be able to do QUERY on; only necessary for non-basic attributes.
CANONICALIZE *dn*	Return the canonicalized DN.
ENUMERATE [*extra-filter-terms*]	List all provisioned domains; or if some *extra-filter-terms* have been specified, all domains matching the filter.
ENUMERATE -ALIASES [*extra-filter-terms*]	(New in MS 7.4) List all provisioned domains and domain aliases; or if some *extra-filter-terms* have been specified, all domains and domain aliases matching the filter.
EXIT	Exit the utility.
LOCATE DOMAIN *domain*	Locate the specified (by name) domain entry. (Release any previously LOCATEd domain.)
LOCATE BASEDN *dn*	Locate the specified (by base DN) domain entry. (Release any previously LOCATEd domain.)
QUERY *attribute*	Return the value(s) of the specified *attribute* of the previously LOCATEd domain.
QUERY -LOCATION=*n* *attribute*	Return the *n*th value of the specified *attribute* of the previously LOCATEd domain. (Recall that the LDAP specification makes no guarantees regarding the order in which attributes and values are returned, so the result of such a command may vary even while the data in LDAP remains the same.)
QUIT	Quit the utility.
RELEASE	Release the presently LOCATEd domain entry.
SHOW	Show the (previously LOCATEd) domain.
USER ADD *dn*	Add the specified user to the utility's in-memory hash table of users belonging to the previously LOCATEd domain.

`USER DELETE` *dn*	Delete the specified user from the utility's in-memory hash table of users belonging to the previously `LOCATE`d domain.
`USER DNADMIN` *dn*	Check whether or not the specified (by DN) user is an admin. (In particular, an admin user entry will have the object classes `groupOfUniqueNames` and `inetMailAdministrator`.)
`USER IDADMIN` *id*	Check whether or not the specified (by ID) user is an admin. (In particular, an admin user entry will have the object classes `groupOfUniqueNames` and `inetMailAdministrator`.)
`USER TEST` *dn*	Check whether the specified user is in the utility's in-memory hash table of users belonging to the previously `LOCATE`d domain.
`VERIFY`	Check the provisioning of domains in LDAP, potentially returning various errors if any error (as regards MTA requirements for domain provisioning) is found.

71.26.4 Error messages

Below are listed possible warnings and errors that may be reported. Note that "errors" are reported when a domain is defined in a way not suitable for MTA use as an addressing or routing domain; however, if a domain is defined in LDAP for use by some component(s) other than (not including) Messaging Server, then it is not *necessarily* an error to have a domain defined in a fashion inappropriate for MTA domain usage. That is, reports of "errors" regarding domain definitions must be interpreted in light of a site's intended usage of domains: errors reported regarding domains used by the MTA are definitely problems, but "errors" reported regarding other domains must be interpreted in light of such a domain's intended usage.

`Missing, empty, invalid or duplicate entry`

- **Facility:** General
- **Severity:** Error
- **Explanation:** Problem with a domain entry

`Identifier too long`

- **Facility:** General (*e.g.*, `imsimta test -domain_map LOCATE` *`domain-name`*)
- **Severity:** Error
- **Explanation:** Specified domain name was too long.

`Attribute not listed`

- **Facility:** General (*e.g.*, `imsimta test -domain_map QUERY` *`attribute-name`*)
- **Severity:** Error
- **Explanation:** Specified attribute not found in domain entry.

`LDAP error`

- **Facility:** General
- **Severity:** Error
- **Explanation:** Trouble communicating with LDAP.

`Bad schema level`

- **Facility:** General
- **Severity:** Error
- **Explanation:** Schema mode other than 1 or 2 specified.

`%DMAP-E-ALIASANDBASEDN, Domain alias entry '`*`domain-name`*`' also contains a base DN`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The supposed domain alias entry has an `inetDomainBaseDN` LDAP attribute (in the case of the MTA, more precisely, whatever LDAP attribute is named by the `ldap_domain_attr_basedn` MTA option), which is an attribute that ought only to be present on an actual domain entry, not on a domain alias entry.
- **User Action:** If this domain entry is truly intended as a domain alias entry, then remove the `inetDomainBaseDN` LDAP attribute from it (while also making sure that the real domain entry does have an `inetDomainBaseDN` attribute present). If, however, this domain entry ought instead to be its own, real domain entry, then remove the LDAP attributes (such as `aliasedObjectName`) that are setting it up as a domain alias entry.

`%DMAP-E-ALIASTOOLONG, Domain alias '`*`domain-name`*`' in entry with DN '`*`DN-string`*`' is too long`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain entry has a `associatedDomain` LDAP attribute (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_attr_domain2_schema2` MTA option) with a value longer than 1024 characters.
- **User Action:** Shorten the `associatedDomain` value.

`%DMAP-E-BASEDNTOOLONG, Base DN '`*`DN-string`*`' in base entry for domain '`*`domain-name`*`' is too long`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** Schema 2. This domain entry has no explicit `inetDomainBaseDN` (optional in Schema 2) and the DN of this domain entry, when converted to a string, is longer than 1024 characters.
- **User Action:** One possibility is to add an explicit `inetDomainBaseDN` attribute (or more precisely, for the MTA add whatever attribute is named by the `ldap_domain_attr_basedn` MTA option) specifying where users/groups in this domain are stored in the DIT; while this may work around this immediate issue, note that if the DNs in your DIT are long enough to trigger this error, then you may encounter other processing difficulties due to the "long" DNs. Alternatively, shorten the DNs for your domains; this may require some restructuring of your domain names or DIT, but lead to more satisfactory processing in general.

`%DMAP-E-BASEDNTOOLONG, Base DN pointer '`*`DN-string`*`' in entry for domain '`*`domain-name`*`' is too long`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetDomainBaseDN` value (for the MTA, more precisely the value of whatever attribute is named by the `ldap_domain_attr_basedn` MTA option) is longer than 1024 characters.
- **User Action:** Shorten the `inetDomainBaseDN`, restructuring the data in the directory, if necessary.

`%DMAP-E-CANONICAL, Overlapping domains '`*`domain-name-1`*`' and '`*`domain-name-2`*`' defined by entries '`*`DN-1`*`' and '`*`DN-2`*`' have different canonical domains '`*`canon-name-1`*`' and '`*`canon-name-2`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain name usage is not consistent: domains with overlapping sets of users have conflicting "canonical" domain names.
- **User Action:** Check the structuring of the domains, especially the use of the `inetDomainBaseDN` and `inetCanonicalDomainName` LDAP domain attributes (or more precisely, the attributes named by the `ldap_domain_attr_basedn` and `ldap_domain_attr_canonical` MTA options).

`%DMAP-E-CANONICALINVALID, Canonical domain '`*`canon-name`*`' defined/referenced by domain entry with DN '`*`dn-string`*`' is syntactically invalid`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetCanonicalDomainName` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_canonical` MTA option) has a value which is not syntactically valid as a domain name.
- **User Action:** Ensure that the canonical domain name is a syntactically valid, fully-qualified, Internet domain name. (Note that a not-fully-qualified domain name, a "short form" host name, will instead result in the error below: `%DMAP-E-CANONICALSHORTFORM`.)

`%DMAP-E-CANONICALSHORTFORM, Short form canonical domain '`*`domain-name`*`' defined/reference by domain entry with DN '`*`dn-string`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetCanonicalDomainName` (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_canonical` MTA option) has its value set to a "short form" host name value rather than a fully qualified domain name.
- **User Action:** Change the value to be a fully qualified domain name. There are some places where it is acceptable to use a "short form" host name, but the canonical domain name should be a fully qualified domain name.

`%DMAP-E-CANONICALTOOLONG, Canonical name '`*`canon-name`*`' in entry for domain '`*`domain-name`*`' is too long`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetCanonicalDomainName` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_canonical` MTA option) has a value longer than 1024 characters.
- **User Action:** Shorten the canonical domain name.

`%DMAP-E-CANTCONVDCDN, Cannot convert alias pointer DN '`*`aliasedObjectName-value`*`' for domain alias `*`domain-name`*` to domain name`

- **Facility:** `dmap_verify`
- **Severity:** Error

- **Explanation:** The DN specified by the `aliasedObjectName` LDAP attribute (in the case of the MTA, more precisely, whatever LDAP attribute is named by the `ldap_domain_attr_alias` MTA option), could not be processed to produce a domain name; presumably it was not of a *organization* sort of form.
- **User Action:** Check that the `aliasedObjectName` value is correctly specified.

`%DMAP-E-CANTCONVDCDN, Cannot convert DN '`*`dn-string`*`' in DC tree to domain name`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** A failure occurred attempting to convert the domain's DN into a domain name.
- **User Action:** Check the LDAP domain structure.

`%DMAP-E-CANTEXTALIAS, Empty alias pointer attribute in '`*`domain-name`*`' domain alias entry`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The supposed domain alias entry has no `aliasedObjectName` LDAP attribute (in the case of the MTA, more precisely, none of whatever LDAP attribute is named by the `ldap_domain_attr_alias` MTA option).
- **User Action:** Add an `aliasedObjectName` attribute pointing to a "real" domain entry.

`%DMAP-F-CANTGETDN, Cannot obtain DN of domain entry, directory error`

- **Facility:** `dmap_verify` routine (called by, *e.g.*, `imsimta test -domain_map verify`)
- **Severity:** Fatal
- **Explanation:** An unexpected error: something "wrong" has happened with the data in the LDAP directory, or with retrieval and basic handling of that data.
- **User action:** There may be something seriously wrong with the LDAP directory. Verify that the LDAP directory is operating and its data is uncorrupted.

`%DMAP-W-DISALLOWEDATTR, Domain '`*`domain-name`*`' has a disallowed attribute '`*`attr-name`*`' with value '`*`string`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning

- **Explanation:** The `iplanet-am-user-account-life` domain attribute is present (and has a value set).
- **User Action:** Remove the attribute.

`%DMAP-E-DNTOOLONG, Domain entry DN beginning with '`*`dn-string`*`' is too long`

- **Facility:** `dmap_verify` routine (called by, *e.g.,* `imsimta test -domain_map verify`)
- **Severity:** Error
- **Explanation:** The DN for this domain is longer than the maximum permitted DN length (1024 characters).
- **User action:** If this domain is one that it is desired that the MTA make use of (say in addressing or routing), then change its DN to something shorter; this may require some restructuring of data and entries in LDAP.

`%DMAP-E-DOMAINALIASINVALID, Domain alias '`*`string`*`' defined/referenced by domain entry with DN '`*`DN-string`*`' is syntactically invalid`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** A domain alias, whether found via `aliasedObjectName` pointer in Schema 1 mode, or set as a `associatedDomain` in Schema 2 mode, (see the `ldap_domain_attr_alias` or `ldap_attr_domain2_schema2` MTA options) is not a syntactically valid domain name.
- **User Action:** Correct the domain alias.

`%DMAP-E-DOMAININVALID, Domain name '`*`domain-name`*`' defined/referenced by domain entry with DN '`*`DN-string`*`' is syntactically invalid`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain name (whether constructed from the "`dc`" chunks in the `DN` in Schema 1 mode, or found in the `sunPreferredDomain` attribute in Schema 2 mode) is not syntactically valid.
- **User Action:** Correct the domain entry.

`%DMAP-E-DOMAINMULTDEF, Domain '`*`domain-name`*`' multiply defined by entries with DNs '`*`DN-string-1`*`' and '`*`DN-string-2`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** Two different domain entries in Schema 2 mode both claim the same domain name, that is, have the same value in either of their respective `sunPreferredDomain` or `associatedDomain` LDAP attributes (or for the MTA, more precisely whatever LDAP attributes are named by the `ldap_attr_domain1_schema2` and `ldap_attr_domain2_schema2` MTA options).
- **User Action:** Correct the domain entries so that each domain entry has its own, unique, domain name (its own, unique, `sunPreferredDomain` value) and so that different domain entries do not claim the same domain alias (same `associatedDomain` value). Note that it is perfectly normal and legitimate to have domain aliases, and overlapping domain name usage is also legitimate and supported -- but there are correct ways to provision such usage. A domain alias is configured in Schema 2 via use of the `associatedDomain` LDAP attribute on the real domain entry. Overlapping domain name usage is configured by use of distinct domain entries that specify and control overlap via `inetDomainBaseDN` and `inetCanonicalDomain` LDAP attributes.

`%DMAP-E-DOMAINTOOLONG, Domain '`*`domain-name`*`' in entry with DN '`*`DN-string`*`' is too long`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain entry's (Schema 2 mode) domain name, that is, the value in its `sunPreferredDomain` LDAP attribute (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_attr_domain1_schema2` MTA option), is longer than 1024 characters.
- **User Action:** Shorten the `sunPreferredDomain` value so that it is at most 1024 characters long.

`%DMAP-E-DOMAINUNDEF, Domain name '`*`domain-name`*`' referenced by domain entry with DN '`*`DN-string`*`' never defined`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain name usage is not consistent; an undefined domain name has been found.
- **User Action:** Check the list of domains that can be properly located in the directory (see for instance the `ENUMERATE` command of `imsimta test -domain_map`), and compare with the list of domains you desire to have defined: look for any "orphan" (undefined/not located) domain name.

`%DMAP-W-EMPAPPSTAT, Domain '`*`domain-name`*`' has an empty application`

`status`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `mailDomainStatus` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_mail_status` MTA option) is present but with no value set.
- **User Action:** The MTA interprets a missing value as meaning "`inactive`", but a valid value should be set (especially as non-MTA components may have other behaviors upon failing to find a valid value).

`%DMAP-W-EMPDEFMAILHOST, Domain '`*`domain-name`*`' has an empty mailHost default`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The domain LDAP attribute named by the `ldap_domain_attr_default_mailhost` MTA option (not set by default) is present, but has no value set.
- **User Action:** Set a valid value, or remove the attribute.

`%DMAP-W-EMPDISALLOWED, Domain '`*`domain-name`*`' has an empty disallowed attribute`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `iplanet-am-user-account-life` domain attribute is present, but with no value set.
- **User Action:** Remove the attribute.

`%DMAP-W-EMPDOMSTAT, Domain '`*`domain-name`*`' base node at `*`DN-string`*` has an empty domain status`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** Schema 2. The domain entry has no `inetDomainStatus` attribute, or such an attribute has no value set. (For MTA purposes, more precisely the attribute named by the `ldap_domain_attr_status` MTA option is not present or is present but with no value set.) Processing will continue after this warning; indeed the MTA treats such a missing attribute or empty value as equivalent to a setting of `active`.

- **User Action:** Consider setting `inetDomainStatus` to a value of `active` for clarity (and to avoid this warning).

`%DMAP-E-EMPTYBASEDN, Domain '`*`domain-name`*`' has an empty base DN`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain entry has an `inetDomainBaseDN` attribute (or for the MTA, more precisely whatever attribute is named by the `ldap_domain_attr_basedn` MTA option), but with no value set for that attribute.
- **User Action:** In Schema 1 mode, this attribute must be present and must have a value: set a value for the attribute. In Schema 2 mode, this presence of this attribute is optional, but if present it must have a value: either remove the attribute entirely (if the user entries are all located directly under the domain entry), or set a correct value for the attribute.

`%DMAP-E-EMPTYCANONICAL, Domain '`*`domain-name`*`' has an empty canonical name`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetCanonicalDomainName` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_canonical` MTA option) is present but does not have a value.
- **User Action:** Either remove the attribute, or set a valid value for it.

`%DMAP-W-EMPUIDSEP, Domain '`*`domain-name`*`' has an empty UID separator attribute`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `domainUidSeparator` domain attribute (or more precisely, the LDAP attribute named by the `ldap_domain_attr_uid_separator` MTA option) is present, but with no value set.
- **User Action:** Set a valid value, or remove the attribute. Note that while the MTA doesn't directly use the value, the authentication library code *does* make use of the attribute's value.

`%DMAP-F-INTDEFERROR, Internal defined flag error on domain '`*`domain-name`*`', aborting`

- **Facility:** `dmap_verify`

- **Severity:** Fatal
- **Explanation:** The code encountered a problem with its internal hashed storage of domain names, possibly a coding error.
- **User Action:** Report this error to Oracle.

`%DMAP-F-INTHASHERROR, Internal hash error, aborting`

- **Facility:** `dmap_verify`
- **Severity:** Fatal
- **Explanation:** The code encountered a problem with its internal hashed storage of domain names or domain base DNs, possibly a coding error.
- **User Action:** Report this error to Oracle.

`%DMAP-F-INTHASHERROR, Internal tree structure error, aborting`

- **Facility:** `dmap_verify`
- **Severity:** Fatal
- **Explanation:** The code encountered a problem with its domain tree structure, possibly a coding error.
- **User Action:** Report this error to Oracle.

`%DMAP-F-INTTREESTRUCTERROR, Internal tree structure error, aborting`

- **Facility:** `dmap_verify`
- **Severity:** Fatal
- **Explanation:** The code encountered a problem with its domain tree structure, possibly a coding error.
- **User Action:** Report this error to Oracle.

`%DMAP-W-INVALIDAPPSTAT, Application status '`*`string`*`' for domain '`*`domain-name`*`' is invalid`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `mailDomainStatus` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_mail_status` MTA option) has an invalid value set: a value which is none of `active`, `inactive`, `overquota`, `removed`, `unused`, `nonlocal`, `disabled`, `hold`, or `deleted`.

- **User Action:** Set a valid value. Although the MTA will interpret invalid values as meaning "`inactive`", it is unwise to rely on that effect when other components with other behavior may also interpret (and interpret differently) the attribute's value.

`%DMAP-E-INVALIDBASEDN, Base DN pointer '`*`DN-string`*`' in entry for domain '`*`domain-name`*`' is not a valid DN`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetDomainBaseDN` domain attribute's value, (for MTA purposes, the value of whatever attribute is named by the `ldap_domain_attr_basedn` MTA option), does not parse properly as a DN.
- **User Action:** Check for syntax errors in the value of `inetDomainBaseDN`.

`%DMAP-E-INVALIDDEFMAILHOST, Default mailHost '`*`host-string`*`' for domain '`*`domain-name`*`' is invalid`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain LDAP attribute named by the `ldap_domain_attr_default_mailhost` MTA option (not set by default) is set to a value which is not a syntactically valid domain name.
- **User Action:** Specify a syntactically valid (and, one hopes, semantically correct) domain name as the value.

`%DMAP-W-INVALIDDOMSTAT, Domain status '`*`status`*`' for domain '`*`domain-name`*`' base node at `*`DN-string`*` is invalid`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** Schema 2. The `inetDomainStatus` value (for MTA purposes, the value of whatever attribute is named by `ldap_domain_attr_status`) was not valid: it was something other than `active`, `inactive`, or `deleted`. Processing will continue after this warning.
- **User Action:** If this is a domain intended for mail use by the MTA, set a valid value for `inetDomainStatus`; however, if this domain is not going to be used for mail purposes, this warning may not be of concern. (Note that "switching" of the meaning of `inetDomainStatus` and `inetDomainMailStatus` is supported by the MTA, by "switching" the values of `ldap_domain_attr_status` and `ldap_domain_attr_mail_status`, with the attribute named by `ldap_domain_attr_mail_status`, normally `inetDomainMailStatus`, allowing

additional supported values. So other than a simply outright "wrong" value for `inetDomainStatus`, another possibility is that the meanings and hence values were "switched" in a prior configuration.)

`%DMAP-W-INVALIDUIDSEP, Domain '`*`domain-name`*`' has an empty UID separator attribute`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `domainUidSeparator` domain attribute (or more precisely, the LDAP attribute named by the `ldap_domain_attr_uid_separator` MTA option) is set to a value containing invalid characters.
- **User Action:** Set a valid value. Note that while the MTA doesn't directly use the value, the authentication library code *does* make use of the attribute's value.

`%DMAP-W-MULTDOMAINNAMES, Domain entry with DN '`*`DN-string`*`' has multiple domain names, used value '`*`name-1`*`' ignored '`*`name-2`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The domain entry has multiple values of the `sunPreferredDomain` LDAP attribute (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_attr_domain1_schema2` MTA option). Processing will continue---but this is presumably a provisioning error.
- **User Action:** Remove the extraneous `sunPreferredDomain` values.

`%DMAP-W-MULTIAPPSTAT, Multivalued application status in entry for domain '`*`domain-name`*`', used value '`*`first-found`*`' ignored '`*`later-found`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `mailDomainStatus` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_mail_status` MTA option) has multiple values set.
- **User Action:** Remove the extraneous values: this should be a single-valued attribute. Note that as LDAP leaves unspecified the order in which values are returned, having multiple values set will result in unpredictable/unreliable results.

`%DMAP-W-MULTIBASEDN, Multivalued base DN pointer in entry for domain '`*`domain-name`*`', used value '`*`DN-string-1`*`' ignored '`*`DN-string-2`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The domain entry has two (or more) `inetDomainBaseDN` values (or for the MTA, more precisely multiple values of whatever LDAP attribute is named by the `ldap_domain_attr_basedn` MTA option). Processing continues, but a warning is issued since this is presumably a provisioning error.
- **User Action:** Remove all but one `inetDomainBaseDN` value, as only one value will be used. Note that if user/group entries for the domain are currently scattered under multiple locations in the DIT, attempting to set multiple `inetDomainBaseDN` values to point to those multiple locations will not work---instead, one `inetDomainBaseDN` must be specified above *all* the user/group entries for the domain, restructing the DIT if necessary to achieve this.

`%DMAP-E-MULTICANONICAL, Multivalued canonical name in entry for domain '`*`domain-name`*`', used value '`*`first-found`*`' ignored '`*`later-found`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The `inetCanonicalDomainName` domain attribute (or more precisely, the LDAP domain attribute named by the `ldap_domain_attr_canonical` MTA option) has multiple values set.
- **User Action:** It is a provisioning error to have multiple canonical domain names set: remove the extraneous value(s). Note that the LDAP specification does not define which value will be returned "first", and indeed the order may vary from LDAP query to LDAP query. So although processing will continue after such an error, the resulting effect during operation may be unreliable, if varying canonical names are returned.

`%DMAP-E-MULTIDEFMAILHOST, Multivalued mailhost default in entry for domain '`*`domain-name`*`', used value '`*`value-0`*`' ignored '`*`value-1`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** The domain LDAP attribute named by the `ldap_domain_attr_default_mailhost` MTA option (not set by default) has multiple values set.
- **User Action:** Remove the extraneous values.

`%DMAP-W-MULTIDOMSTAT, Multivalued domain status in entry for domain '`*`domain-name`*`' base node at `*`DN-string`*`, used value '`*`status-0`*`' ignored '`*`status-1`*`'`

- **Facility:** `dmap_verify`

- **Severity:** Warning
- **Explanation:** Schema 2. Multiple `inetDomainStatus` values (for MTA purposes, multiple values for whatever attribute is named by `ldap_domain_attr_status`) were found. Processing will continue after this warning, with whatever value LDAP happened to return first being used.
- **User Action:** Remove all but one (correct) value for `inetDomainStatus`.

`%DMAP-W-MULTIUIDSEP, Multivalued UID separator in entry for domain '`*`domain-name`*`', used value '`*`value-0`*`' ignored '`*`value-1`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `domainUidSeparator` domain attribute (or more precisely, the LDAP attribute named by the `ldap_domain_attr_uid_separator` MTA option) has multiple values.
- **User Action:** Remove the extraneous values. Recall that as LDAP makes no guarantees regarding the order in which values are returned, having multiple values can lead to unpredictable/unreliable results. Note that while the MTA doesn't directly use the value, the authentication library code *does* make use of the attribute's value.

`%DMAP-W-MULTIVALIAS, Multivalued alias pointer in entry for domain alias '`*`domain-name`*`', used value '`*`DN-string-0`*`', ignored '`*`DN-string-x`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The supposed domain alias entry has multiple values of the `aliasedObjectName` LDAP attribute (in the case of the MTA, more precisely, multiple values for whatever LDAP attribute is named by the `ldap_domain_attr_alias` MTA option). All values other than the "first" that happened to be reported on this LDAP query will be ignored---but recall that LDAP makes no guarantee as to the order in which attributes are returned, so such ordering and hence which value is used may vary between different queries! Since getting inconsistent effects, varying with the order in which the LDAP directory happened to return results, is unlikely to be desirable, this is a warning of what can be presumed to be a provisioning error, though processing will continue using the "first" value reported by LDAP.
- **User Action:** Remove all the redundant or extraneous `aliasedObjectName` values, so that only the one correct value remains in the domain alias entry, and so that consistent processing will occur.

`%DMAP-E-NOBASEDN, Domain '`*`domain-name`*`' has no base DN`

- **Facility:** `dmap_verify`

- **Severity:** Error
- **Explanation:** Schema 1. This domain entry has no explicit `inetDomainBaseDN`, (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_domain_attr_basedn` MTA option); such an attribute is required in Schema 1.
- **User Action:** Add an `inetDomainBaseDN` attribute specifying where users/groups in this domain are stored in the DIT.

`%DMAP-E-NOBASEDN, Domain '`*`domain-name`*`' has an empty base DN`

- **Facility:** `dmap_verify`
- **Severity:** Error
- **Explanation:** Schema 1 or Schema 2. This domain entry has an `inetDomainBaseDN` LDAP attribute, (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_domain_attr_basedn` MTA option) with no value set.
- **User Action:** Specify a value for the `inetDomainBaseDN` LDAP attribute specifying where users/groups in this domain are stored in the DIT; or in Schema 2 *only* another alternative is to remove the `inetDomainBaseDN` attribute entirely, to indicate that users and groups in this domain will be located directly under this domain entry in the DIT.

`%DMAP-W-NOBASEDNNODE, Base DN pointer '`*`DN-string`*`' in entry for domain '`*`domain-name`*`' doesn't point at anything`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The `inetDomainBaseDN` value (for the MTA, more precisely the value of whatever attribute is named by the `ldap_domain_attr_basedn` MTA option) corresponds to no actual node in the directory. Processing will continue after this warning.
- **User Action:** If the domain in question is intended for current use, then this is a warning of some form of provisioning error: either the `inetDomainBaseDN` value is incorrect, or the data supposed to be in the directory is missing or in the "wrong" location. However, if this is a "place-holder" domain entry intended for later use, but not yet fully provisioned (in particular not yet provisioned with any users), then this warning may not be of concern at this time. Decide whether the domain in question is intended for current use, and if so, ensure that its `inetDomainBaseDN` value points to the intended base DN under which its users are, or will be, located.

`%DMAP-E-NODOMAINNAME, Domain entry with DN '`*`DN-string`*`' does not have a domain name`

- **Facility:** `dmap_verify`
- **Severity:** Error

- **Explanation:** Schema 2. The domain entry does not have a `sunPreferredDomain` LDAP attribute (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_attr_domain1_schema2` MTA option).
- **User Action:** Add the domain name to the entry; that is, add a `sunPreferredDomain` LDAP attribute specifying the desired domain name.

`%DMAP-W-NODOMAINNAME, Domain entry with DN '`*`DN-string`*`' has a blank domain alias`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** Schema 2. The domain entry has a `associatedDomain` LDAP attribute (or for the MTA, more precisely whatever LDAP attribute is named by the `ldap_attr_domain2_schema2` MTA option) with no value. Processing continues, but a warning is issued since this is presumably a provisioning error.
- **User Action:** Either set a valid value in `associatedDomain`, or remove the attribute.

`%DMAP-W-NOENTRIES, No domain entries found, aborting`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** No domain entries were found.
- **User Action:** This is the expected output if pointing to an LDAP directory that does not contain Schema 1 or Schema 2 compatible domain entries. However, if the MTA has been configured to consult an LDAP directory that does (supposedly) contain valid domain entries, then something is wrong with basic configuration: *e.g.*, MTA configuration for what LDAP directory to consult is incorrect, or the MTA cannot see the domain entries (authentication problems, or ACLs preventing access), or the MTA has not been configured properly for where in the DIT to look to find domain entries.

`%DMAP-W-SHORTFORMDEFMAILHOST, Default mailHost '`*`host-string`*`' for domain '`*`domain-name`*`' is a shortform name`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The domain LDAP attribute named by the `ldap_domain_attr_default_mailhost` MTA option (not set by default) is set to a short form hostname value, rather than to a fully qualified domain name.
- **User Action:** Use of fully qualified domain names is strongly recommended: change value to be in fully qualified form.

`%DMAP-W-SHORTFORMDOMAIN, Short form domain name '`*`domain-name`*`' defined/referenced by domain entry with DN '`*`DN-string`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** The domain name (whether constructed from the "`dc`" chunks in the `DN` in Schema 1 mode, or found in the `sunPreferredDomain` attribute in Schema 2 mode) is a "short form" host name, rather than a fully qualified domain name.
- **User Action:** Use of fully qualified domain names is *strongly* recommended; strongly consider revisiting domain name usage and switching to use of fully qualified domain names.

`%DMAP-W-SHORTFORMDOMAINALIAS, Short form domain alias '`*`string`*`' is defined/referenced by domain entry with DN '`*`DN-string`*`'`

- **Facility:** `dmap_verify`
- **Severity:** Warning
- **Explanation:** A domain alias has been specified in "short form" host name form, rather than as aa fully qualified domain name.
- **User Action:** Use of fully qualified domain names is *strongly* recommended; strongly consider revisiting domain name usage and switching to use of fully qualified domain names.

71.27 test -eightbit utility

Test whether a file contains eight bit material.

71.27.1 Syntax

```
imsimta test -eightbit filename
```

Table 71.21 imsimta test -eightbit Command Switches

Switch	Default
-performance	-noperformance
-summary	-nosummary
-number	-number
-mode=n	-mode=0

71.27.2 Parameters

71.27.2.1 *filename*

The name of the file to inspect for eight bit material.

71.27.3 Description

Check whether or not a file contains eight bit material.

The output of this utility will say either:

```
File contains one or more 8-bit chars
```

or

```
File contains no 8-bit chars
```

In -noperformance -nosummary operation (the default), the output will also show which line(s) contained eight bit material, with line numbers (unless -nonumber has been specified).

71.27.4 Switches

71.27.4.1 -performance, -noperformance (default)

When -performance is specified, the utility reads the entire file into memory and then scans the in-memory copy of the file. -performance operation may be faster, but does not allow for outputting which line(s) contained eight bit material. -performance may not be specified if -summary is specified.

71.27.4.2 -summary, -nosummary (default)

Specifying -summary disables outputting which specific lines contained eight bit material. -nosummary (meaning that such lines are output) is the default. -summary may not be explicitly specified if -performance is specified; -performance always causes summary-like output.

71.27.4.3 -number (default), -nonumber

-number (the default) means that when outputting lines containing eight bit material (which is done in -nosummary -noperformance operation), to prefix those lines with line numbers. -nonumber turns off the line numbering.

71.27.4.4 -mode=*n*

The default is -mode=0.

71.27.5 Examples

```
# imsimta test -eightbit IMTA_LIB:locale/en/return_error.txt
File contains no 8-bit chars
# imsimta test -eightbit IMTA_LIB:locale/fr/return_error.txt
     1 Des erreurs de traitement se sont produites lors de la distribution?:
     4 Le traitement de la distribution s'est poursuivi malgr? ces erreurs.
File contains one or more 8-bit chars
```

The above two commands show example output for a file containing no eight bit characters, compared to a file that does contain eight bit characters. (Note that unless you have gone to special work on a UNIX system to properly display eight bit characters, any eight bit character is likely to appear as a question mark, as shown in this example.)

71.28 test -expression utility

Test an expression, *e.g.*, a Sieve filter or configuration recipe.

71.28.1 Syntax

```
imsimta test -expression
```

Table 71.22 imsimta test -expression Command Switches

Switch	Default
`-statement=`*n*	`-statement=1`
`-multiple`	`-multiple`
`-block`	`-noblock`
`-symbols`	`-nosymbols`
`-uav=`*n*	`-uav=1`
`-input=`*filename*	*None*
`-output=`*filename*	*None*
`-parseonly`	*None*
`-string=`*n*	`-string=65535`
`-iterations=`*n*	`-iterations=`*max_sieve_match_iterations*
`-list=`*n*	`-list=`*max_sieve_list_size*
`-debug[=`*n*`]`	`-debug=1`
`-mm`	*See text*
`-xc`	*See text*
`-message=`*filename*	*None*
`-required`	`-required`
`-from=`*address*	`-from="`*postmaster-address*`"`
`-to=`*recipient-address*	`-noto`
`-envid=`*id*	`-noenvid`
`-system`	`-nosystem`
`-mtpriority=`*n*	`-mtpriority=0`
`-sender=`*sender*	`-nosender`
`-username=`*username*	`-nousername`
`-utf8`	`-noutf8`
`-source=`*source-channel-name*	`-source=l`
`-rsecret=`*recall-secret*	`-norsecret`

71.28.1.1 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 group specified by the imta_world_group MTA Tailor option value), in order to have any hold or capture actions applied by a Sieve filter be shown in the utility's output.

71.28.1.2 Prompts

Table 71.23 imsimta test -exression Prompts

Prompt	Value
Expression:	*expression*

71.28.2 Description

Test an expression, such as a Sieve filter or a configuration recipe. test -expression -mm tests message enqueue functions, such as Sieve filters (with the -message switch). test -expression -xc tests recipe syntax operating on a Unified Configuration. Testing consists of parsing the expression, which converts it to an internal compiled form, and then evaluating the compiled form. The -parseonly switch can be used to disable the evaluation step.

Basic arithmetic and string operators are supported in expressions; and when variables may be used, that is, if the -symbols switch has been used, then additional operations upon variables are supported. For a list of the supported operators (with one exception), see the recipe language's list of Operators in Order of Precedence. The one difference between the operator support for the recipe language, *vs*. for Sieve filters, is that since in Sieve syntax square brackets represent lists, indexing into a string in Sieve must be performed using parentheses rather than square brackets. So for instance a Sieve script:

```
require "fileinto";
str := "12folderXYZ";
substr := str(3,9);
fileinto substr;
```

is equivalent to '`fileinto "folder";`'. (See Variable indices under Recipe language for further discussion of indexing into strings and lists.)

Without the -mm switch, test -expression tests symbol table functions (in general, comprising various string functions). The symbol table functions available include those shown in Symbol table functions. (Note that this set of functions supported by imsimta test -expression and by Sieve filters does *not* include all of the Recipe language functions discussed under Recipe language; the recipe language includes support for additional functions useful in configuration recipes.)

Conditional expressions may be used, having the form:

test ? *then-result* : *else-result*

However, "if..." tests of the form

if *test* {*then-result*} else {*else-result*}

are only allowed in certain contexts; in particular, they are not allowed when only basic expressions are permitted, as in mapping table expression substitutions.

Table 71.24 Symbol table functions

Function	Description
abs(*i***)**	Return the absolute value of *i*.
allof(*i1*[,*i2*...]**)**	Returns a nonzero value if all of *i1*, *i2*, ... are nonzero; returns 0 otherwise.
any(*s1*,*s2***)**	Return 2 if any character in *s1* appears as the first character of *s2*; return 0 otherwise.
anyof(*i1*[,*i2*...]**)**	Returns a nonzero value if any of *i1*, *i2*, ... are nonzero, returns 0 if all are zero.
bal(*c1*,*c2*,*c3*,*s***)**	Scan *s* looking for an occurence of a character in *c1* that is balanced with respect to *c2* and *c3*. Returns the position of the first balanced *c3* character in *s* if one is found, length(s)+1 if no *c3* character is found but the string as a whole is balanced, or 0 if the string isn't balanced.
center(*s1*,*n*,*s2***)**	Return a string that has string *s1*'s last character at position *n*, padding on the left with the character or characters from string *s2*. *s2* may be omitted, in which case space characters are used to pad out the string.
check8(*s***)**	Return 1 if string *s* contains any eight bit data; return 0 otherwise.
check8(*s*,*n***)**	If *n* is 0, it is ignored and the function returns 0 or 1 according to whether the string *s* contains any eight bit data. If *n* is any positive integer, then return the string *s* with any eight bit characters replaced by the question mark character, "?".
chr(*i1*[,*i2*...]**)**	Returns a string containing successive characters with decimal values *i1*,*i2*, ...
datetime	Return the current date and time, in RFC 822/RFC 1123 date-time string format, *e.g.*, "Fri, 10 Oct 2008 15:40:02 -0700 (PDT)"
decode *[:e] s*	Decodes the string *s* from the specified encoding *:e*. The *:e* nonpositional parameter must be one of :base64, :base85, :hex, :idn, or :quotedprintable. The default is :hex.
defined(*s***)**	Returns 1 if *s* is defined as a variable; return 0 otherwise.
encode *[:e] s*	Encodes the string *s* into the specified encoding *:e*. The *:e* nonpositional parameter must be one of :base64, :base85, :hex, :idn, :param, :quotedprintable, or :url. The default is :hex.
find(*s1*,*s2*[,*i*,*j*]**)**	Returns the position of the first occurence of *s1* in *s2*[*i*:*j*]. The entire string is searched if i and j are omitted.
find(*s*,*l*[,*i*,*j*]**)**	Returns the position of the first list element from *l*[*i*:*j*] that matches *s*. The entire list is searched if i and j are omitted.
genid	Return a unique id string, such as that used for generating Message-id's or message queue file names.
integer(*e***)**	Converts *e* to an integer. If *e* is already an integer it is returned unchanged; if *e* is a string it is read as a sequence of ASCII digits. If *e* is a list it must contain one element and is treated in the same way a string would be.

lcase(*e***)**	Converts any upper case characters in *e* to lower case. If *e* is a number it is converted to a string.
left(*s1***,***i***[,***s2***])**	Returns leftmost *i* characters of *s1*. If *i* is greater than length(*s1*) the result is padded with *s2*. As much of *s2* as is necessary will be used; if *s2* is too short it will be used multiple times. *s2* defaults to a space if it is omitted.
left(*l1***,***i***[,***l2***])**	Returns leftmost *i* elements of *l1*. If *i* is greater than length(*l1*) the result is padded with *l2*. As much of *l2* as is necessary will be used; if *l2* is too short it will be used multiple times. *l2* defaults to one empty list element if it is omitted.
length(*s***)**	Returns the number of 8-bit characters in the string *s*.
length(*l***)**	Returns the number of elements in the list *l*.
list(*s***[,***n***])**	Returns a list *n* elements long with each element equal to the string *s*. If omitted *n* defaults to 1.
list(*l***[,***n***])**	Returns a list consisting of *n* copies of list *l*. If omitted *n* defaults to 1.
map(*s1***,***s2***,***s3***)**	Returns a string obtained by mapping characters of *s1* that occur in *s2* into corresponding characters in *s3*. Characters that don't appear in *s2* are unchanged.
max(*i***,***j***[,...])**	Returns the largest element in a set of integers.
match(*r***,***s***)**	Returns 1 (true) if the regular expression *r* matches a substring of string *s*, 0 (false) otherwise. Note that the pattern *r* 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.
max(*s1***,***s2***[,...])**	Returns the largest element in a set of strings.
min(*i***,***j***[,...])**	Returns the smallest element in a set of integers.
min(*s1***,***s2***[,...])**	Returns the smallest element in a set of strings.
repl(*s***,***j***)**	Returns a string consisting of *j* concatenations of *s*.
repl(*l***,***j***)**	Returns a list consisting of *j* concatenations of *l*.
reverse(*s***)**	Reverses all the characters in *s* and returns the result.
reverse(*l***)**	Reverses all the elements in *l* and returns the result.
right(*s1***,***i***[,***s2***])**	Returns rightmost *i* characters of *s1*. If *i* is greater than length(*s1*) the result is padded with *s2*. As much of *s2* as is necessary will be used; if *s2* is too short it will be used multiple times. *s2* defaults to a space if it is omitted.
right(*l1***,***i***[,***l2***])**	Returns rightmost *i* elements of *l1*. If *i* is greater than length(*l1*) the result is padded with *l2*. As much of *l2* as is necessary will be used; if *l2* is too short it will be used multiple times. *l2* defaults to one empty list element if it is omitted.
sign(*i***)**	Returns -1 if *i* < 0, 0 if *i* = 0, +1 if *i* > 0.
sort(*l1***[,***i***[,***l2***]])**	Sorts the elements of *l1* to be in ascending order if *i* <> 0 and descending order if *i* = 0. *i* defaults to 1 if it is omitted. If *l2* is present its elements are shifted in the same way as elements in *l1* are shifted.

split(*s*[,*c*[,*i*]]**)**	Produces a list of elements consisting of pieces of *s* delineated by characters in *c*. If omitted *c* defaults to a comma. If *i* is 0 or 1, zero length elements are preserved; if *i* is 2, they are not. If omitted *i* defaults to 1.
split(*l*[,*c*[,*i*]]**)**	Produces a list of elements consisting of pieces of elements of *l* delineated by characters in *c*. If omitted *c* defaults to a comma. If *i* is 0, boundaries between the original elements aren't preserved and zero length elements can be output; if *i* is 1, boundaries are preserved and zero length elements can be output; if *i* is 2, boundaries aren't preserved and zero length elements are omitted. If omitted *i* defaults to 1.
string(*e***)**	Converts *e* to a string. If *e* is already a string it is returned unchanged. If *e* is an integer it is converted to a string. If *e* is a list, the string that results from concatenating the elements of *e* is returned.
string(*l*,*s***)**	Converts the list *l* to a string, inserting the string *s* between each pair of elements of *l*. So for instance `string(["a","cd","e"],"01")` would return the string `"a01cd01e"`.
string(*i*[,*j*[,*k*]]**)**	(New in MS 6.3 are the optional second and third arguments.) Converts the integer *i* to a string, optionally padding with zeros (on the left) so that the length of the string is *j*, and optionally outputting the result in the radix specified by *k*. So for instance `string(15,8,2)` returns `00001111`.
trim(*s*[,*c*]**)**	Returns *s* with any trailing characters found in *c* removed. *c* defaults to space and tab if omitted.
trim(*l*[,*c*]**)**	Returns *l* with any trailing characters found in *c* removed from each element. *c* defaults to space and tab if omitted.
type(*e***)**	Returns `"integer"` if *e* evaluates to an integer, `"string"` if *e* evaluates to a string, and `"list"` if *e* evaluates to a list.
ucase(*e***)**	Converts any lower case characters in *e* to upper case. If *e* is a number it is converted to a string.
Directory function	Description
gettag(*x***)**	If *x* evalutes to a string, returns the tag character for that string. If *x* evaluates to a list, returns the default and per-element tags as a string. An error will be returned if *x* evalutes to anything other than a string or list.
settag(*x*,*y***)**	Sets the tag or tags of *x* to *y*. *x* may be either a string or a list. If *x* is a string, then *y* must be a one character string. If *x* is a list, then *y* must be a string of length `length(`*x*`)+1` and the first character of *y* must be a space.

The expression to test may be read from a file via the `-input` switch. If `-input` is not specified, then the utility enters an interactive loop, prompting for expressions to test with an "`Expression:`" prompt; the utility will exit when CTRL/D (UNIX) is entered.

By default, each line of expression is evaluated independently (separately). The `-block` switch may be used to tell the utility to evaluate the entire block of input expression(s) at once, as a whole; thus in particular, `-block` is useful when evaluating a multi-line Sieve filter.

New in MS 7.0u2, `imsimta test -expression -mm` will return a nonzero status in the event of either a parse or evaluation error.

71.28.3 Switches

71.28.3.1 `-block`, `-noblock` (default)

By default (`-noblock`), input is evaluated one line at time. With the `-block` switch, evaluation is postponed until the entire input file has been read (if the `-input` switch is used), or until CTRL/D is entered (if `-input` is not used and expressions are being entered interactively). Thus in particular `-block` allows use of multi-line Sieve constructs.

71.28.3.2 `-debug[=`*n*`]`, `-nodebug` (default)

The `test -expression` utility is capable of outputting additional, detailed information about the internal steps of its operation. The `-debug` switch enables this output. Specifying `-debug` is equivalent to specifying `-debug=1`; greater amounts of debug information can be requested by specifying `-debug=`*n* for values of *n* up to 10.

Note that a debug level of 2 or more enables output of strings specified via Sieve "`debug`" actions. Note that a debug level of 3 or more enables output (via "`mmc_output_line(`*x*`) header line:`" debug output lines) of the header lines of the message file being processed, while a debug level of 4 or more enables output not only of the header lines, but also of the message body lines handled in memory (via "`mmc_output_line(`*x*`) internal body line:`" debug output lines) of the message file being processed, and a debug level of 5 or more enables output of the message body lines handled via an external file for "large" message bodies (via "`mmc_output_line(`*x*`) external body line:`" debug output lines). `-debug=1` is the default.

New in MS 7.0u2, `-nodebug` turns off the dump of the expression.

71.28.3.3 `-envid=`*id*, `-noenvid` (default)

(New in MS 7.0u3.) `-envid` is only available when `-mm` and `-message` are specified. `-envid` sets the envelope ENVID value for use with the Sieve envelope-dsn extension for envelope ENVID tests. `-noenvid` is the default.

71.28.3.4 `-from=`*return-address*, `-to=`*recipient-address*, `-noto` (default)

(`-from` new in MS 6.2; `-to` and `-noto` new in MS 6.3) `-from` and `-to` are only available when `-mm` and `-message` are specified. `-from` sets the return address for use in Sieve envelope From comparisons. If this switch is not used, the default postmaster address is assumed (see the `return_address` MTA option).

To specify an empty (null) envelope From, the command wants to see `-from=""`--- however shell quoting, *e.g.*, `-from=\"\"`, may be required in order to get such an argument through the shell.

`-to` sets the recipient address for use in Sieve envelope To comparisons. If this switch is not used, or if `-noto` is explicitly specified, then no recipient address is set.

71.28.3.5 -input=*filename*

The -input switch tells the utility to read its input from the specified file, rather than prompting for and reading input interactively. Note that when using -input to enter a Sieve filter file, one usually also wants to use -block (so that multi-line Sieve constructs can be handled).

71.28.3.6 -iterations=*n*

The -iterations switch may be used to specify the maximum number of iterations permitted in the internal code implementing :matches, overriding (for this command execution) any setting of the max_sieve_match_iterations MTA option. (Note that the default value of max_sieve_match_iterations is 1,000,000,000.)

71.28.3.7 -list=*n*

The -list switch may be used to specify the maximum size of list permitted, overriding (for this command execution) any setting of the max_sieve_list_size MTA option. (Note that the default value of max_sieve_list_size is 64.)

71.28.3.8 -message=*filename*

The -message switch may only be used when the -mm switch is present. It is used to specify an input message to process; this should be an RFC 822 message, not a message file from the MTA's queue area (which contains additional, envelope information beyond the basic RFC 822 message itself).

71.28.3.9 -mm, -xc

The -mm switch means to test using the MTA's normal enqueue code. The -xc switch (new in MS 7.5) means to test recipe language syntax, operating on a Unified Configuration. -mm and -xc may not be specified together.

71.28.3.10 -mtpriority=*n*, -nomtpriority (default)

(New in MS 8.0) -mtpriority takes a required integer argument specifying the initial MT-PRIORITY value. This can affect, for instance, the value of the Sieve environment item vnd.oracle.mtpriority. This may only be specified when -mm has also been specified. Furthermore, it is only relevant when -message has been specified (though it will not result in an error to specify -mtpriority without -message).

71.28.3.11 -multiple (default), -nomultiple

Specifying -nomultiple disables evaluating multiple, semicolon-separated, result-generating, statements per physical line. When -nomultiple is specified, only the first result found when evaluating the line will be returned. -multiple is the default.

71.28.3.12 -output=*filename*

The -output switch tells the utility to output to the specified file, rather than to the terminal screen.

71.28.3.13 -required (default), -norequired

(New in MS 6.2.) The `-required` or `-norequired` switch may only be used when `-mm` is specified. Specifying `-norequired` causes Sieve "`require`" clauses to not be required: any Sieve operation that would normally need an appropriate "`require`" clause in order to be permitted will be permitted even without such a "`require`" clause. In other words, `-norequired` gives an effect as if the Sieve script in question had initially done a

```
require list-of-all-possible-extensions;
```

71.28.3.14 `-rsecret=`*`recall-secret`*, `-norsecret` (default)

(New in MS 8.0) Specify the recall secret. This is only relevant when `-message` has been specified.

71.28.3.15 `-sender=`*`sender-address`*, `-nosender` (default)

(New in MS 8.0.) The `-sender` switch may be used to specify an "authenticated" sender address, as if SMTP AUTH had been used. For instance, the Sieve environment item `vnd.sun.authenticated-sender-address` will be set to the sender value, if specified. This is only relevant when `-message` has been specified.

71.28.3.16 `-source=`*`source-channel-name`*, `-nosource`

(New in MS 8.0) Specify the source channel; the default is `l` (the "L"ocal channel). This is only relevant when `-message` has been specified.

71.28.3.17 `-statement=`*`n`*

Set symbol table statement parsing flags. The argument is a bit-encoded integer, with default value 1. The meanings of the various bits are shown below.

Table 71.25 Statement parsing flags

Bit(s)	Value(s)	Usage
0	1	Allow statements
1	2	Allow loops, including the Sieve "`loop`" construct
2-4	4-28	Debug level (4, 8, 12, ... 28)
6	64	Allow mapped file
7	128	Allow directives
8	256	Enforce "`require`" must be used only at beginning of Sieve
9	512	Allow bracketed lists
10	1024	Return evaluation error details

Bit 0 is the least significant bit.

Note that use of the `-mm` switch sets bit 9 (allow bracketed lists). Use of the `-debug` switch sets bits 2 through 4 (debug level).

71.28.3.18 `-string=`*`n`*

The -string switch may be used to specify the maximum length of string permitted; the default is 65535.

71.28.3.19 -symbols, -nosymbols (default)

The -symbols switch allows the use of variables and symbolic values, and operations may be performed on variables. For instance, symbolic names such as IMTA_TABLE may be used to reference the imta_table directory specification, and assignments may be made to symbolic variable names, such as a := 1, and operators such as those shown in Operators in Order of Precedence of the recipe language may be used. -nosymbols is the default.

71.28.3.20 -system, -nosystem (default)

(New in MS 7.0.5) The -mm switch must be used in order to use -system. If -system is specified with -mm, then the Sieve is treated as a system-level Sieve; if not, then it is treated as a user-level Sieve.

71.28.3.21 -uav=*n*

The -uav switch controls the interpretation of unassigned variables. The default is -uav=1.

Table 71.26 Unassigned variable interpretation

Value	Usage
0	Variables must be predefined; variable creation not allowed
1	Assignment defines variable; no default value
2	Define variable upon first reference; default value ""
3	Define variable upon first reference except in modify operations; default value 0
4	(New in MS 8.0/patch to MS 7.0u5) Same as value 1 (assignment defines variable; no default value), with the difference (appropriate for Sieve usage) don't create a symbol table at parse time

71.28.3.22 -username=*username*, -nousername (default)

(New in MS 8.0) The -username switch may be used to specify a username, relevant when username-based checks are in use. For instance, the Sieve environment item vnd.sun.authenticated-sender-id will be set to the username value, if specified. This is only relevant when -message has been specified.

71.28.3.23 -utf8, -noutf8 (default)

(New in MS 8.0) The -utf8 switch specifies allowing use of internationalized email addresses, as with the SMTPUTF8 SMTP extension (from RFC 6531). -noutf8 is the default.

71.28.4 Examples

```
# ./imsimta test -expression -mm -block -message=msg.txt
```

```
Expression: require "fileinto";
Expression: if header :contains ["Subject"] ["foo"] { fileinto "foo"; }
Expression: if header :contains ["Subject"] ["test"] { discard; }
Expression: CTRL/D
Dump: header:2000111;0  3  1  :contains  1  "Subject"  1  "foo"  if
Dump: 5  ;  fileinto:2000110;0  1  1  "foo"  ;  header:2000111;0  3
Dump: 1  :contains  1  "Subject"  1  "test"  if  3  ;  discard:2000107;0
Dump: 0
Result: 1
Filter result: [ discard ]
```

The above command sequence tests the result of various Sieve filter command lines on a message file, `msg.txt`, that for the purposes of this example is assumed to have a Subject: header line that contains the word "test" but does not contain the word "foo".

```
# ./imsimta test -expression -mm -block -message=msg.txt -from=\"\"
Expression: require ["envelope","fileinto"];
Expression: if envelope :all :is "from" "" { fileinto "notifications"; }
Expression: CTRL/D
Dump: envelope:4000115;0  4  1  :all  1  :is  1  "from"  1  ""  if  5
Dump: ; fileinto:4000114;0  1  1  "notifications"
Result: "notifications" ' '
Filter result: [ fileinto "notifications" ]
```

The above command sequence tests the result of a Sieve filter script that files all notification messages (messages with empty envelope From) to a folder named "`notifications`", when that Sieve filter is presented with some message and when the envelope From is set to be empty via the `-from` switch.

```
# ./imsimta test -expression -mm -statement=3 -block -message=msg.txt"
Expression: loop { discard; exitif (true); }
Expression: CTRL/D
Dump: loop  5  discard;0  0  ;  1  exitif
Result: 1
Filter result: [ discard ]
```

The above example shows use of the new-in-MS-7.0.5 "`loop`" construct in a Sieve filter. Note that `-statement=3` must be specified on the command line in order to use a "`loop`" construct in this utility.

71.29 test -hash utility

Test generating a hash of a string or of file contents.

71.29.1 Syntax

```
imsimta test -hash hash-name [file-name]
```

Table 71.27 imsimta test -hash Command Switches

Switch	Default
`-string=`*`string`*	`-nostring`
`-repetitions=`*`n`*	*None*
`-key=`*`key`*	*None*

71.29.2 Parameters

71.29.2.1 *hash-name*

Required parameter specifying the hash function to use. Valid values are `MD2`, `MD4`, `MD5`, `SHA1`, `SHA256`, `SHA512`, `MD128`, or `MD160`.

71.29.2.2 *file-name*

Optional parameter specifying the name of a file whose contents to input to the hash function. The `-string` switch may not be used if a file is specified.

71.29.3 Description

The `imsimta test -hash` utility generates a hash of string or file input. This may be of interest when testing the MTA's ability to generate message hashes, a feature useful for message archiving purposes.

71.29.4 Switches

71.29.4.1 `-string=`*`string`*, `-nostring` (default)

`-string` may not be specified if a file-name parameter has been supplied.

71.29.4.2 `-key=`*`key`*

May only be used when a string is being hashed (`-string`), not when a file is specified.

71.29.4.3 `-repetitions=`*`n`*

May only be used when a string is being hashed (`-string`), not when a file is specified.

71.29.5 Examples

```
# ./imsimta test -hash -string="just a test" SHA1
Hash value: E6F36746CCBA42C288ACF906E636BB278EAEB7E8
```

In the above example, a hash of the string "`just a test`" is generated using the SHA-1 hash function.

71.30 test -header utility

Test message header processing.

71.30.1 Syntax

```
imsimta test -header
```

Table 71.28 imsimta test -header Command Switches

Switch	Default
`-length=`*`n`*	`-length=80`
`-increment=`*`n`*	`-increment=20`
`-alignment=`*`n`*	`-alignment=0`
`-options=`*`filename`*	*None*
`-push`	`-nopush`
`-apply`	`-noapply`
`-folds`	`-nofolds`
`-blank`	`-noblank`
`-digits`	`-nodigits`
`-trim`	`-trim`
`-keeporder`	`-nokeeporder`
`-input=`*`filename`*	*None*
`-output=`*`filename`*	*None*
`-postscript`	`-nopostscript`
`-dump`	`-nodump`

71.30.2 Parameters

None.

71.30.3 Description

The `imsimta test -header` utility parses header lines, optionally performing certain types of header trimming, and outputs the processed header lines (including, prior to MS 7.0u4, a PostScript version). The utility may either read header lines from a file (`-input=`*`filename`*) or from the command line (in that case prompting with `header>`). The output takes the general form:

```
--- Output with options ---

parsed> header-line-1
parsed> header-line-2
...
```

```
--- PS ---

PostScript-output

--- After decode/encode and no options ---

parsed> header-line-1
parsed> header-line-2
...
```

though as of MS 7.0u4, the PostScript output is optional and will not be included unless `-postscript` is explicitly specified. The "`Output with options`" section of the output is what results from performing any relevant header trimming specified via the `-options` switch. The "`After decode/encode and no options`" section of the output shows canonicalization effects on header field names and header field values, but does not include effects of `-options` header trimming unless `-apply` has been specified.

71.30.4 Switches

71.30.4.1 `-alignment=`*n*

Specify the alignment point for header lines, analogous to the `headerlabelalignment` channel option; the default is 0, which causes header lines not to be aligned.

71.30.4.2 `-apply`, `-noapply` (default)

`-apply` causes application of any relevant header trimming directives specified via the `-options=`*`filename`* switch, applying such trimming to the underlying header structure. Note that not all trim options make sense to apply to the underlying header structure, as some trim options relate to output, not to the underlying header storage. So for instance trim options such as RELABEL, and LINELENGTH, are not relevant (and will not occur) with `-apply`. `-apply` is only relevant when `-options` has been specified (`-apply` has no effect unless `-options` has been specified), so `-apply` cannot sensibly be combined with `-keeporder`; (as of MS 7.0u4, indeed the combination is an error).

71.30.4.3 `-blank`, `-noblank` (default)

The default is `-noblank`.

71.30.4.4 `-digits`, `-nodigits` (default)

The default is `-nodigits`.

71.30.4.5 `-dump`, `-nodump` (default)

(New in MS 7.0u4) When `-dump` is specified, the output includes, prior to the "After applying options" output:

```
--- Dump of header ---
dump of header structure
```

71.30.4.6 -folds, -nofolds (default)

The default -nofolds switch corresponds to the headerfoldremove channel option, while the -folds switch corresponds to the headerfoldpreserve channel option.

71.30.4.7 -increment=*n*

Specify the increment used when attempting to fold header lines, analogous to the headerlineincrement channel option; the default is 20.

71.30.4.8 -keeporder, -nokeeporder (default)

Specifying -keeporder causes -options to be ignored, and forces -noapply. (As of MS 7.0u4, attempting to specify either -options or -apply with -keeporder is an error.)

71.30.4.9 -linelength=*n*

Specify the length at which to wrap header lines, analogous to the headerlinelength channel option and the LINELENGTH header trimming option; the default is 80.

71.30.4.10 -options=*filename*, -nooptions (default)

The -options switch specifies a header trimming option file to open, read, and parse. Options relevant to output of header lines will be shown when -apply is not specified; if -apply is specified then only those options affecting the underlying header structure (header structure storage) will take effect. (In particular, with -apply set, only ADD, FILL, and PRECEDENCE will take effect. In contrast, with the default -noapply, then options such as RELABEL and LINELENGTH will also take effect.) -options can not be combined with -keeporder; (indeed, as of MS 7.0u4 specifying both is an error).

71.30.4.11 -postscript, -nopostscript (default)

(New in MS 7.0u4) -nopostscript (the default as of MS 7.0u4) disables the outputting of a PostScript version of the header lines. Specifying -postscript causes the PostScript version to be output; prior to MS 7.0u4, this was not controllable with the PostScript version always being output.

71.30.4.12 -push, -nopush (default)

Specifying -push causes the MTA to use a different way of reading header information. The results with -push *vs.* -nopush should always be identical; report to Oracle if differences are seen. -push is ignored if -input has been specified; (indeed, as of MS 7.0u4 attempting to specify both is an error). -nopush is the default.

71.30.4.13 -trim (default), -notrim

Control whether or not to trim trailing white space off header lines. The default is -trim, meaning to perform such white space trimming.

71.30.5 Examples

```
# ./imsimta test -header
header> CoNtEnT-TrAnsFeR-eNcOdInG: bAsE64
header>
--- After applying options ---

parsed> Content-transfer-encoding: bAsE64

--- PS ---

/Courier-Bold findfont 10 scalefont
/Courier findfont 10 scalefont
/SHOW_HEADERS where
{pop (Content-transfer-encoding: bAsE64) SHOW_HEADERS}
{35 720 moveto exch dup setfont (Content-transfer-encoding:) show exch dup
setfo
nt ( bAsE64) show} ifelse
pop pop

--- After decode/encode and no options ---

parsed> Content-transfer-encoding: BASE64
header> Ctrl-D
#
```

The above example from an older version (when the PostScript was included by default) shows the canonicalization of the Content-transfer-encoding: header line field name, and field value.

```
# ./imsimta test -header -options=IMTA_TABLE:return_header.opt -input=sample_headers.txt

--- Output with options ---

parsed> Message-id: <01NYWMSLISS600H4Y@domain.com>
parsed> Date: Mon, 14 Mar 2011 14:18:50 -0700 (PDT)
parsed> From: John Doe <jdoe@domain.com>
parsed> To: Jane Brown <jbrown@domain.com>
parsed> Subject: New report

--- After decode/encode and no options ---

parsed> Received: from localhost by host.domain.com
parsed>  (Oracle Communications Messaging Exchange Server 7u5-2.03 64bit)
parsed>  (built Feb  6 2011) with ESMTP id <0G8P00B01ZFNUV@domain.com>
parsed>  for jbrown@domain.com (ORCPT jbrown@domain.com); Mon,
parsed>  14 Mar 2011 15:47:47 -0800 (PDT)
parsed> Date: Mon, 14 Mar 2011 15:47:47 -0800 (PDT)
parsed> From: John Doe <jdoe@domain.com>
parsed> Subject: New report
parsed> To: Jane Brown <jbrown@domain.com>
parsed> Message-id: <01NYWMSLISS600H4Y@domain.com>
parsed> MIME-version: 1.0
parsed> Content-type: MULTIPART/MIXED
#
```

The above example shows applying the default header trimming file, `return_header.opt`, used when constructing DSNs; see Sample distributed `return_header.opt` file. Note that such trimming has removed the Received:, MIME-version:, and Content-type: header lines, and moved the Message-Id: header line above the other retained header lines.

71.31 test -mapping utility

Test an MTA mapping table.

71.31.1 Syntax

```
imsimta test -mapping [input-string]
```

Table 71.29 imsimta test -mapping Command Switches

Switch	Default
`-context`	`-context`
`-debug`	`-nodebug`
`-destination_channel=`*`channel`*	`-nodestination_channel`
`-flags=`*`list-of-characters`*	`-noflags`
`-image_file`	`-image_file`
`-mapping_file=`*`file-spec`*	`-mapping_file=IMTA_TABLE:mappings`
`-option_file=`*`file-spec`*	`-option_file=IMTA_TABLE:option.dat`
`-table=`*`table-name`*	*None*
`-reload`	`-noreload`
`-input=`*`file-spec`*	`-input=stdin`
`-originator=`*`address`*	`-nooriginator`
`-output=`*`file-spec`*	`-output=stdout`
`-sources`	`-sources`
`-source_channel=`*`channel`*	`-nosource_channel`

71.31.1.1 Prompts

Table 71.30 imsimta test -mapping Prompts

Prompt	Value
`Enter table name:`	*`table-name`*

71.31.2 Parameters

71.31.2.1 *`input-string`*

Optional input string to run through the mapping.

71.31.3 Description

`imsimta test -mapping` may be used to test the behavior of an MTA mapping table. The string resulting from mapping an input string will be output along with a list of any metacharacters specified in the output string.

If an input string is supplied on the command line, then only the result of mapping that input string will be output. If no input string is specified, then `imsimta test -mapping` will enter a loop, prompting for an input string, mapping that string, and prompting again for another input string. `imsimta test -mapping` will exit when a CTRL/D (UNIX) is entered.

Note that this utility is testing only the formal syntax and formal result of a mapping table; it is not testing what the mapping table *means* (the effect of the mapping table) for actual MTA operation. In particular, for testing the effect of the `FROM_ACCESS` mapping table or recipient address `*_ACCESS` mapping tables, see instead the `test -rewrite` utility.

Or in the other (less semantic, more syntactic) direction, note that the matching of wildcards in mapping table patterns (left hand sides of entries) may be tested in detail via the `test -match` utility, which may be of particular interest when using "complex" wildcards such as character "glob" matches, or IP address prefix matches.

71.31.4 Switches

71.31.4.1 `-context` (default), `-nocontext`

(New in MS 7.0) The `-context` switch is the default, and causes the mapping table to be probed as if operating in a normal, MTA usage, application (with normal MTA initialization having been performed). `-nocontext` causes the mapping table to be probed as if operating in a mode without normal MTA initialization, and hence may be useful for testing mapping table operation in non-MTA contexts.

71.31.4.2 `-debug`, `-nodebug` (default)

The mapping test process is capable of producing additional, detailed information about the sequence of steps in the mapping probe(s) and match(es). Such information is particularly likely to be of interest (useful) with iterative mapping tables, or mapping tables that involve callouts; that is, such debug information is most likely to be of interest when exploring implications of `$C`, `$L`, or `$R` metacharacter use, or effects of callouts such as to other mapping tables, to the general database, to LDAP, or to Oracle or site-supplied routines. The `-debug` switch enables this output; it is disabled by default.

71.31.4.3 `-destination_channel=`*`channel`*

(New in MS 8.1.0.1) The `-destination_channel` switch is used to specify a destination channel when the MTA is initialized for writing. The source channel is used if no desitnation channel is specified. See the documentation on the `-source_channel` switch below for additional information.

71.31.4.4 `-flags=`*`list-of-characters`*, `-noflags` (default)

The `-flags` switch is used to specify particular flags to set during the mapping testing; for instance, the E (envelope), B (header/body), R (backwards pointing), or I (message id) flags when testing a `REVERSE` mapping. Multiple flags may be specified by concatenating them, *e.g.*, `-flags=BR`.

71.31.4.5 `-image_file` (default), `-noimage_file`

When the `-image_file` switch is specified (the default), the MTA will load the compiled configuration file *CONFIGROOT*`/advanced/config_data` (located prior to MS 7.0 via

the imta_config_data MTA Tailor option). When -noimage_file is specified, then imsimta test -mapping will unconditionally ignore any compiled mapping information and instead read mapping information directly from the named MTA mapping groups (Unified Configuration) or the mappings file (legacy configuration).

71.31.4.6 -input=*filename*

(New in MS 8.0) By default, imsimta test -mapping takes input from stdin. The -input switch may be used to specify a different source for input.

71.31.4.7 -mapping_file=*filename*

This switch instructs imsimta test -mapping to use the specified mapping file rather than the default MTA mapping file, *CONFIGROOT*/mappings (located prior to MS 7.0 via the imta_mapping_file MTA Tailor option). This switch has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any MTA mapping values (Unified Configuration) or physical mapping file (legacy configuration); this switch also has no effect when an XML configuration is in use.

71.31.4.8 -option_file=*filename*, -nooption_file

This switch instructs imsimta test -mapping to use the specified option file rather than the default MTA option file, *CONFIGROOT*/option.dat (prior to MS 7.0 located via the imta_option_file MTA Tailor option). This switch has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any MTA options whether from mta.*option-name* (Unified Configuration) or from a physical option file (legacy configuration). Use of the switch -nooption_file will prevent the MTA option file from being read in when there is no compiled configuration. These switches have no effect when an XML configuration is in use.

71.31.4.9 -originator=*address*

(New in MS 8.1.0.1) The -originator switch is used to specify the envelope from (MAIL FROM) address when the MTA is initialized for writing. The value "mail@fom" is used if no originator address is specified. See the documentation on the -source_channel switch below for additional information.

71.31.4.10 -output=*output_file_spec*

(New in MS 8.0) By default, imsimta test -mapping writes output to stdout. The -output switch may be used to direct the output of imsimta test -mapping elsewhere.

71.31.4.11 -reload, -noreload (default)

The -reload switch may be used to tell the utility to reread MTA mapping group options (Unified Configuration) or the mappings file (legacy configuration) if the mapping table name specified is not found cached in memory.

71.31.4.12 -source_channel=*channel*

(New in MS 8.1.0.1) Mappings do not depend on the MTA being initialized for writing a message, but some mapping plugins do. In order to test such plugins, the -source_channel can be used to cause the enqueue logic to be initialized using the specified channel. Also see

the -originator and -destination_channel switches, which can be used to specify the envelope from (MAIL FROM) address and destination channel, respectively.

71.31.4.13 -sources (default), -nosources

(New in MS 8.0) The -nosources switch may be used to suppress the (new in MS 8.0) output of segment sources.

71.31.4.14 -table=*table-name*

This switch specifies the name of the mapping table to test. If this switch is not specified, then imsimta test -mapping will prompt for the name of a table to use.

71.31.5 Examples

```
# imsimta test -mapping -noimage_file -mapping_file=IMTA_TABLE:mac_mappings.sample
Enter table name: MAC-TO-MIME-CONTENT-TYPES
Input string: BINHEX|7344424e|4d535744|Test.doc
Output string: APPLICATION/MSWORD
Output flags: [0, 'Y' (89)]
Input string: ^D
#
```

In the above UNIX example, the sample MAC-TO-MIME-CONTENT-TYPES mapping is tested. The -mapping_file switch is used to select the mapping file mac_mappings.sample instead of the default mapping file.

```
# imsimta test -mapping -flags=A -table=X-FLAGS
Input string: testing
Output string: testing plus A flag
Output flags: [0, 'Y' (89)]
Input string: ^D
# imsimta test -mapping -flags=B -table=X-FLAGS
Input string: next
Output string: next without A flag
Output flags: [0, 'Y' (89)]
Input string: ^D
```

The above example assumes that the mappings file contains a mapping table:

```
X-FLAGS

   *              $C$:A$0$ plus$ A$ flag$Y$E
   *              $;A$0$ without$ A$ flag$Y
```

That X-FLAGS mapping table outputs the original input string plus suffix string " plus A flag" if the input flag "A" was present, or outputs the original input string plus suffix string " without A flag" if the input flag "A" was not set. In the above test -mapping utility output, it can be seen that input flags, if any, do *not* carry over to being present amongst the set of output flags: an input flag set via the -flags switch does not carry over to appear amongst the output flags in the "Output flags:" portion of the output. Input flags and

output flags are distinct and separately stored. Also note how the entry with the `$:A` test has that test "wrapped" with the "`$C`...`$E`"; this is to cause the mapping process to "continue on" if that entry (in particular the test for the presence of the input flag "`A`") failed, but "end" if the test succeeded. In this particular example, the "`$E`" could have been omitted since the only subsequent entry explicitly checked for the opposite ("`A`" flag not present) condition. However wrapping with "`$C`...`$E`" is better practice, and is commonly used in entries containing tests.

71.32 test -match utility

Test a mapping wildcard pattern.

71.32.1 Syntax

```
imsimta test -match
```

71.32.2 Parameters

None.

71.32.3 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 exitted with a CTRL/D (UNIX).

71.32.4 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.

71.33 test -mime utility

Test a message's MIME structure.

71.33.1 Syntax

```
imsimta test -mime [input-file [output-file]]
```

Table 71.31 imsimta test -mime Command Switches

Switch	Default
-line	-noline
-convert	-noconvert
-header	-noheader
-thurman	-nothurman
-security	-nosecurity
-rotate[=*n*]	-rotate=0 or -norotate
-debug	-nodebug
-mparts=*n*	-mparts=0
-mlevels=*n*	-mlevels=0
-message_handling=*keyword*	-message_handling=KEEP
-multipart_handling=*keyword*	-multipart_handling=KEEP
-format=*keyword*	*See text*
-encoding=*keyword*	-encoding=NONE
-mode=*keyword*	
-length=*n*	-length=0
-eightbit	*See text*
-charset=*charset-name*	-charset=US-ASCII
-new_charset=*charset-name*	*See text*
-number	-number
-level	*See text*
-describe	*See text*
-pmaximum=*n*	-pmaximum=1024
-nmaximum=*n*	-nmaximum=128
-pformat=*n*	-pformat=0
-iencoding	-iencoding
-iemessage	-iemessage
-iemultipart	-iemultipart
-channel=*channel-name*	-channel=*scan_channel-value*
-scan=*sieve-expression*	*None*

71.33.1.1 Restrictions

None.

71.33.1.2 Prompts

Table 71.32 imsimta test -mime Prompts

Prompt	Value
`Input file:`	*`file-spec`*
`Output file:`	*`file-spec`*

71.33.2 Parameters

71.33.2.1 *`input-file`*

Parameter specifying the name of the input message file.

71.33.2.2 *`output-file`*

Optional parameter specifying a file to which to write the utility's output. If no output file is specified, the output is written to the terminal.

71.33.3 Description

`imsimta test -mime` may be used to test and analyze, or convert, a message's MIME structure. When used with `-convert` or `-archive`, the utility's output is respectively the result of converting, or constructing an archive version of, the message. When used with `-scan` and `-channel`, the utility runs the specified Sieve filter and/or channel configured spam/virus filtering. Otherwise, the output is an analyzed (and, depending on the switches specified, optionally somewhat annotated) copy of the original message.

71.33.4 Switches

71.33.4.1 `-archive=`*`archive-channel-name`*, `-noarchive` (default)

(New in MS 6.3.) `-archive` cannot be used with `-convert` or `-line`. The default is `-noarchive`. Specifying `-archive=`*`archive-channel-name`* causes the MTA to write an archive file per the configuration of the specified archive channel.

71.33.4.2 `-channel=`*`channel-name`*, `-nochannel` (default)

(New in 7.0.5) The `-channel` switch may be used when `-scan` is used; it must be used if it is desired to engage spam/virus filter package scanning. If `-channel` is specified without an argument, then the channel defaults to the setting of the `scan_channel` MTA option, or the local channel if `scan_channel` is not set. Note that spam/virus filter package scanning is only engaged if `-channel` is explicitly specified.

71.33.4.3 -convert, -noconvert (default)

-convert cannot be used with -archive or -line.

71.33.4.4 -describe, -nodescribe

-describe only matters when -line has been specified. Setting -new_charset overrides any setting of -describe or -nodescribe: depending upon whether the specified new charset is one known to the MTA the describe flag is either set, or cleared, respectively. In the absence of -new_charset, setting either -prefix or -suffix will force the describe flag to be set. When the describe flag is set, due either to one of the above-mentioned override effects, or otherwise via explicit -describe switch use, then a Content-transfer-encoding: header line and a Content-MD5: header line can be added to message parts.

71.33.4.5 -eightbit, -noeightbit

Specifying -eightbit sets the "eight always" handling; specifying -noeightbit sets the "seven only" handling. Note that the default behavior corresponds to neither keyword, corresponding instead to "eight negotiate" handling.

71.33.4.6 -encoding=*keyword*

The default is -encoding=NONE. Valid values for the encoding keyword are: NONE, QUOTED_PRINTABLE, BASE32, OBASE64, BASE64, CBASE64, DBASE64, HEXADECIMAL, UUENCODE, CUUENCODE, DUUENCODE, PATHWORKS, BINHEX, BTOA, BASE85, or CDATA.

71.33.4.7 -format=*keyword*

-format cannot be used unless -line is used. When -line is used, the default is -format=MIME. The valid keyword arguments for -format are: MIME, RFC1154, MAILWORKS, NEXT, HEADER_SET, and STREAM_DECODE.

71.33.4.8 -iencoding (default), -noiencoding

New in MS 6.3. The default is -iencoding, which means to interpret (decode) message content whose encoding is described by the non-standard Encoding: header line. Specifying -noiencoding means to ignore any such Encoding: header line; that is, not perform decoding. These switches are thus analogues of the channel options interpretencoding and ignoreencoding.

71.33.4.9 -iemessage (default), -noiemessage

New in MS 6.3. The default is -iemessage, which means to interpret (decode) encodings of MIME message parts; note that such encodings are illegal, but may sometimes be encountered in messages from incompliant software. Specifying -noiemessage means to ignore any such Content-transfer-encoding: illegally present on a message part. These switches are thus analogues of the channel options interpretmessageencoding and ignoremessageencoding.

71.33.4.10 -iemultipart (default), -noiemultipart

New in MS 6.3. The default is -iemultipart, which means to interpret (decode) encodings of MIME multiparts; note that such encodings are illegal, but may sometimes

be encountered in messages from incompliant software. Specifying `-noiemultipart` means to ignore any such Content-transfer-encoding: illegally present on a multipart. These switches are thus analogues of the channel options `interpretmultipartencoding` and `ignoremultipartencoding`.

71.33.4.11 `-line`, `-noline` (default)

`-line` cannot be used with `-convert` or `-archive`.

71.33.4.12 `-message_handling=`*keyword*

Valid arguments for `-message_handling` are: `KEEP`, `TOP`, `BOTTOM` or `END`, `DELETE`, or `MERGE`. The default is `-message_handling=KEEP`.

71.33.4.13 `-mode=`*keyword*

Valid arguments for `-mode` are: `CRATTRIBUTE`, `LFATTRIBUTE`, `CRLFATTRIBUTE`, `BLOCK`, `RECORD`, `TEXT[=n]`, `POSTSCRIPT`, `ENRICHED`, `FLOWED`, `HTML`, `DOUBLEAPPLE`, `SINGLEAPPLE`, `MACBINARY`, `BINHEX`, `VIRUSSCAN`. `-mode=text` means `-mode=text=80`.

71.33.4.14 `-number` (default), `-nonumber`

The `-number` switch is ignored when used with `-archive` or `-capture`. When used with `-line` (or with none of `-line`, `-archive`, or `-capture`), it tells the utility to output initial table heading lines prior to its output analyzing the message structure. Combined with `-line`, `-number` causes table heading lines of:

```
Line # ?
------ -
```

Or if none of `-line`, `-archive`, or `-capture` is specified, then `-number` causes table heading lines of:

```
Line # Level Count ?
------ ----- ----- -
```

Note that `-number` is the default; specifying `-nonumber` disables this table heading output.

71.33.4.15 `-rotate=`*n*, `-norotate` (default)

`-rotate` is equivalent to `-rotate=13`.

71.33.4.16 `-nmaximum=`*n*

(New in MS 6.0) Specify a maximum allowed length for the NAME parameter and for the FILENAME parameter on the Content-type: and Content-disposition: MIME header lines, respectively; longer parameters will be truncated. This switch is thus an analogue of the `nameparameterlengthlimit` channel option. The default is 128.

Note that other, general parameters on the Content-type: and Content-disposition: MIME header lines are controlled instead via the `-pmaximum` switch. (The reason why such lengths

are of interest, and why these switches exist, is due to the history in certain popular e-mail clients of security problems involving buffer overruns.)

71.33.4.17 -number (default), -nonumber

Specifying -nonumber turns off the display of line numbers in the output. -number is the default.

71.33.4.18 -pformat=*n*

The default is 0.

71.33.4.19 -pmaximum=*n*

(New in MS 6.0) Specify a maximum allowed length for general parameters on Content-type: and Content-disposition: MIME header lines; longer parameters will be truncated. This switch is thus an analogue of the parameterlengthlimit channel option. The default is 1024.

Note that the Content-type: NAME parameter and Content-disposition: FILENAME parameter maximum length are controlled separately via the -nmaximum switch. (The reason why such lengths of interest, and these switches exist, is due to the history in certain popular e-mail clients of security problems involving buffer overruns.)

71.33.4.20 -scan=*sieve-expression*, -noscan (default)

(New in MS 7.0.5) The -scan switch specifies a Sieve expression to apply to the input message. This emulates imexpire message scanning. If -channel is specified, then spam/virus filter package scanning can be engaged also. If -channel is not specified, then spam/virus filter package scanning is not engaged, and the channel value (when relevant) defaults to the value of the scan_channel MTA option.

71.33.4.21 -thurman, -nothurman (default)

Specifying -thurman causes the MTA to perform "sniffing" of non-MIME message bodies for UUENCODE or BINHEX "blobs" (akin to application of the thurman channel option), and to pull such "blobs" out into MIME attachments instead.

71.33.5 Examples

```
% imsimta test -mime not-zz.00
Line # Level Count ?
------ ----- ----- -
     1     0     1 H Received: from localhost by elvira.innosoft.com (PMDF V6.0-24 #43970)
     2     0     1 H  with ESMTP id <0G8P00B01ZFNUV@elvira.innosoft.com> for
     3     0     1 H  kristin@elvira.innosoft.com (ORCPT kristin@elvira.innosoft.com); Tue,
     4     0     1 H  13 Feb 2001 15:47:47 -0800 (PST)
     5     0     1 H Date: Tue, 13 Feb 2001 15:47:47 -0800 (PST)
     6     0     1 H From: System Privileged Account <root@elvira.innosoft.com>
     7     0     1 H Subject: test of relaying
     8     0     1 H To: kristin hubner <kristin@elvira.innosoft.com>
     9     0     1 H Message-id: <Pine.SOL.4.21L.0102131547360.15366-100000@elvira.innosoft.com>
    10     0     1 H MIME-version: 1.0
    11     0     1 H Content-type: TEXT/PLAIN; charset=US-ASCII
    12     0     1 D ctype: TEXT
```

```
    13     0     1 D csubtype: PLAIN
    14     0     1 D cparameters:
    15     0     1 D charset=US-ASCII
    16     0     1 B test
    17     0     1 B
% imsimta test -mime -line not-zz.00
Line # ?
------ -
     1 Received: from localhost by elvira.innosoft.com (PMDF V6.0-24 #43970)
     2  with ESMTP id <0G8P00B01ZFNUV@elvira.innosoft.com> for
     3  kristin@elvira.innosoft.com (ORCPT kristin@elvira.innosoft.com); Tue,
     4  13 Feb 2001 15:47:47 -0800 (PST)
     5 Date: Tue, 13 Feb 2001 15:47:47 -0800 (PST)
     6 From: System Privileged Account <root@elvira.innosoft.com>
     7 Subject: test of relaying
     8 To: kristin hubner <kristin@elvira.innosoft.com>
     9 Message-id: <Pine.SOL.4.21L.0102131547360.15366-100000@elvira.innosoft.com>
    10 MIME-version: 1.0
    11 Content-type: TEXT/PLAIN; charset=US-ASCII
    12
    13 test
    14
```

In the above UNIX example, the `not-zz.00` message file contains:

```
Received: from localhost by elvira.innosoft.com (PMDF V6.0-24 #43970)
 with ESMTP id <0G8P00B01ZFNUV@elvira.innosoft.com> for
 kristin@elvira.innosoft.com (ORCPT kristin@elvira.innosoft.com); Tue,
 13 Feb 2001 15:47:47 -0800 (PST)
Date: Tue, 13 Feb 2001 15:47:47 -0800 (PST)
From: System Privileged Account <root@elvira.innosoft.com>
Subject: test of relaying
To: kristin hubner <kristin@elvira.innosoft.com>
Message-id: <Pine.SOL.4.21L.0102131547360.15366-100000@elvira.innosoft.com>
MIME-version: 1.0
Content-type: TEXT/PLAIN; charset=US-ASCII

test
```

71.34 test -rewrite utility

Test address rewriting specified by an MTA configuration; test syntactic validity of an MTA configuration.

71.34.1 Syntax

```
imsimta test -rewrite [test-address]
```

Table 71.33 imsimta test -rewrite Command Switches

Switch	Default
-aby=*value*	-noaby
-additions	-noadditions
-alias_file=*file-spec*	-alias_file=IMTA_TABLE:aliases
-alternate_recipient=*address*	-noalternate_recipient
-by=*value*	-noby
-applicationinfo=*string*	*None*
-channel[=type]	-channel=forward
-check_expansions	-nocheck_expansions
-configuration_file=*file-spec*	-configuration_file=IMTA_TABLE:imta.cnf
-conversion_file=*file-spec*	-conversion_file=IMTA_TABLE:conversions
-database=*database-list*	*See text*
-debug	-nodebug
-delivery_receipt	*See text*
-destination_channel=*channel*	*None*
-ebm	-noebm
-esmtpused	-noesmtpused
-expandlimit=*n*	*None*
-extra_local_channel=*channel*	*None*
-filter	-nofilter
-from=*address*	-from=postmaster@*localhost*
-header	-noheader
-identifiers	-noidentifiers
-image_file[=*file-spec*]	-image_file
-input=*input-file-spec*	-input=stdin
-jacket	-nojacket
-lmtpused	-nolmtpused
-local_alias=*value*	-nolocal_alias
-mapping_file=*file-spec*	-mapping_file=IMTA_TABLE:mappings

-multiple -nomultiple	(see text)
-mtpriority=*n*	-mtpriority=0
-option_file=*file-spec*	-option_file=IMTA_TABLE:option.dat
-output=*output-file-spec*	output=stdout
-password=	-nopassword
-proxyused	-noproxyused
-quotacheck	-quotacheck
-read_receipt	*See text*
-reprocessing	-reprocessing
-restricted=*setting*	-restricted=0
-rrvs=*ISO8601-value*	-norrvs
-saslused	-nosaslused
-sender=*address*	-nosender
-size=*n*	-nosize
-soptin	-nosoptin
-source_channel=*channel*	-source_channel=l
-spares	-nospares
-statistics	-nostatistics
-srs=([domain=*domain*], [secrets=*secrets*], [maxage=*maxage*])	*None*
-system_filter=*file-spec*	-system_filter=IMTA_TABLE:imta.filter
-tag=*tag-list*	-notag
-tlsused	-notlsused
-transportinfo=*string*	*None*
-user=*string*	-user="--USERNAME--"
-utf8	-noutf8
-xml_config=*file-spec*	-xml_config=IMTA_TABLE:config.xml

71.34.1.1 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.

71.34.1.2 Prompts

Table 71.34 imsimta test -rewrite Prompts

Prompt	Value
`Address:`	*test-address*

71.34.2 Parameters

71.34.2.1 *test-address*

Optional parameter specifying one or more (comma-separated) addresses to rewrite.

71.34.3 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 switches 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 canonicalization of specific (local) addresses resulting from address reversal (*i.e.*, testing how local addresses will appear in header lines),
8. testing the effects of address-based `*_ACCESS` mapping tables,
9. testing the effects of posting restrictions such as restrictions on mailing list postings,
10. determining which Sieve filters are applicable for a particular recipient address,
11. testing SRS/MUL encoding and decoding of addresses (if SRS MTA options have been configured).

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 configurat
ion 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.

71.34.4 Switches

71.34.4.1 -aby=*value*, -noaby (default)

(New in MS 8.0) The -aby switch may be used to specify an alternate address deliver by value. -noaby is the default.

71.34.4.2 -additions, -noadditions (default)

(New in MS 8.0) Specifying -additions causes any added prefix or suffix text to be displayed. (That is, -additions causes display of text added via alias options alias_prefix_text or alias_suffix_text, or via mailing list named parameters [PREFIX_TEXT] or [SUFFIX_TEXT], or via LDAP attributes such as mgrpMsgPrefixText or mgrpMsgSuffixText.) -noadditions is the default.

71.34.4.3 -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 IMTA_TABLE:aliases; as of MS 7.0, the alias file is located as *CONFIGROOT*/aliases. The -alias_file switch specifies an alternate file for imsimta test -rewrite to use. This switch 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; this switch also has no effect when an XML configuration is in use.

71.34.4.4 -alternate_recipient=*address*

(New in MS 8.0) Specify an alternate recipient address.

71.34.4.5 -applicationinfo=*string*

This switch 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".

71.34.4.6 -by=*value*, -noby (default)

(New in MS 8.0) The -by switch may be used to specify a deliver by (SMTP DELIVERBY extension BY parameter) value. -noby is the default.

71.34.4.7 -channel*[=type]* (default), -nochannel

This switch controls whether the utility outputs detailed information, *e.g.*, channel flags, regarding the channel an address matches.

As of MS 8.0.2.1, this switch accepts an optional channel type, which controls which of the various channels selected by the rewriting process is displayed. Possible values are forward, backward, source, and destionation. The default is forward, which displays the channel selected by rewriting the input address in the "forward" direction. Note that source and

destination display channel information for the selected source and destination channels, which are not address-dependent.

71.34.4.8 -check_expansions, -nocheck_expansions (default)

This switch 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` switch 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.

71.34.4.9 -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 `IMTA_TABLE:imta.cnf`; as of MS 7.0, the MTA configuration file is located at *CONFIGROOT*`/imta.cnf`. The `-configuration_file` switch 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; this switch also has no effect when an XML configuration is in use.

71.34.4.10 -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 `IMTA_TABLE:conversions`; as of MS 7.0, in legacy configuration the conversion file is located as *CONFIGROOT*`/conversions`; in Unified Configuration, a separate file is not used and instead the `conversions` MTA option stores the conversions. The `-conversion_file` switch specifies an alternate file to use in place of the regular converison 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` switch will prevent the conversion file from being read in when there is no compiled configuration. These switches also have no effect when an XML configuration is in use.

71.34.4.11 -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.

71.34.4.12 -debug, -nodebug (default)

The address rewriting process is capable of producing additional, detailed explanations of what actions are taken and why. The `-debug` switch 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 recipient `*_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.

71.34.4.13 -delivery_receipt, -nodelivery_receipt

The `-delivery_receipt` and `-nodelivery_receipt` switches, 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.

71.34.4.14 -destination_channel=*channel*

The `-destination_channel` switch controls for which destination or target channel `imsimta test -rewrite` rewrites addresses. Some address rewriting is destination channel specific; this switch allows control of the assumed destination channel.

71.34.4.15 -esmtpused, -noesmtpused (default)

(New in MS 6.3.) The `-esmtpused` 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).

71.34.4.16 -expandlimit=*n*

(New in 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.

71.34.4.17 -extra_local_channel=*channel*

(New in MS 7.0u2) The `-extra_local_channel` switch may be used to specify the name of a channel to be treated as if it were the local channel.

71.34.4.18 -filter, -nofilter (default)

The `-filter` switch 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 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 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 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 MTA 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.

Must be superuser or the MTA user in order to display `-filter` output.

71.34.4.19 `-from=`*`address`*`, -nofrom`

The -from switch controls what envelope From address is used for access control probes and mailing list access probes. If this switch 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.

71.34.4.20 -header, -noheader (default)

The -header switch 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 switch), 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.

71.34.4.21 -identifiers, -noidentifiers (default)

(New in MS 7.0.5) The -identifiers switch tells the utility to rewrite the parameter value 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.)

71.34.4.22 -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 IMTA_TABLE:advanced/config_data. As of MS 7.0, the compiled configuration is located as *CONFIGROOT*/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.

71.34.4.23 -input=*input-file-spec*

By default, imsimta test -rewrite takes input from stdin. The -input switch may be used to specify a different source for input.

71.34.4.24 -jacket, -nojacket (default)

Specifying -jacket will result in passing any $I flag checks in the FROM_ACCESS mapping table or recipient address *_ACCESS mapping tables.

71.34.4.25 -lmtpused, -nolmtpused (default)

(New in MS 6.3.) The -lmtpused 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).

71.34.4.26 -local_alias=*value*, -nolocal_alias (default)

This switch 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 switch 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.

71.34.4.27 `-multiple`, `-nomultiple`

(New in 8.0.1.2) Normally the specified address is presented to the enqueue machinery all at once. Specifying `-multiple` causes the argument to be treated as a comma-separated list of addresses; each address will be presented for enqueue processing separately, simulating the effect of multiple RCPT TOs.

The MTA's enqueue facilities handle the submission of multiple addresses at the same time as an extension. Specifying `-nomultiple` disables this capability; multiple addresses will be considered an error.

71.34.4.28 `-mapping_file[=`*`file-spec`*`]`, `-nomapping_file`

If no compiled configuration is being used, then this switch 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 `IMTA_TABLE:mappings`; as of MS 7.0, in legacy configuration the mappings file is located as *CONFIGROOT*`/mappings`; in Unified Configuration, a separate file is not used and instead mappings are stored under named `mapping` groups. These switches 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` switch will prevent the MTA mapping file from being read in when there is no compiled configuration. These switches also have no effect when an XML configuration is in use.

71.34.4.29 `-mtpriority=`*`n`*`, -nomtpriority` (default)

(New in MS 8.0) `-mtpriority` takes a required integer argument specifying the initial MT-PRIORITY value.

71.34.4.30 `-option_file[=`*`filename`*`]`, `-nooption_file`

If no compiled configuration is being used, then the `-option_file` switch 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 `IMTA_TABLE:option.dat`; as of MS 7.0, in legacy configuration the MTA option file is located at *CONFIGROOT*`/option.dat`; in Unified Configuration, a separate file is not used and instead these are MTA options. These switches 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` switch will prevent the MTA option file from being read in when there is no compiled configuration. These switches have no effect when an XML configuration is in use.

71.34.4.31 `-output=`*`output-file-spec`*

By default, `imsimta test -rewrite` writes output to `stdout`. The `-output` switch may be used to direct the output of `imsimta test -rewrite` elsewhere.

71.34.4.32 -password=

Used to specify the password for a password-protected list.

A prompt will appear that allows the password to be entered without echo.

71.34.4.33 -proxyused, -noproxyused (default)

(New in 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).

71.34.4.34 -quotacheck (default), -noquotacheck

(New in MS 8.0.1.2.) The -quotacheck switch controls whether or not the alias expansion process treats an account's overquota status as an error. -quotacheck is the default; -noquotacheck may be used to disable the check. This may be useful in simulating the behavior of temporary failure reenqueue operations, which disable this check.

71.34.4.35 -read_receipt, -noread_receipt

The -read_receipt and -noread_receipt switches, 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.

71.34.4.36 -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").

71.34.4.37 -restricted=*setting*

This switch controls the setting of the restricted flag. By default, this flag has value 0. When set to 1, *i.e.*, -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.

71.34.4.38 -rrvs=*ISO8601-value*, -norrvs

(New in MS 8.0) The -rrvs switch is used to specify an RRVS value (Require-Recipient-Valid-Since: value), in ISO 8601 format. -norrvs is the default.

71.34.4.39 -saslused, -nosaslused (default)

(New in 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).

71.34.4.40 -sender=*address*, -nosender (default)

New in MS 6.2. The -sender switch may be used to set the "authenticated sender" field, for use in FROM_ACCESS mapping table probes.

71.34.4.41 -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.

71.34.4.42 -soptin, -nosoptin (default)

(New in MS 7.0 update 2.) Control whether or not to show per-recipient spamfilter "opt-in".

Must be superuser or the MTA user in order to display -soptin output.

71.34.4.43 -source_channel=*channel*

The -source_channel switch 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, l on UNIX.

71.34.4.44 -spares, -nospares (default)

(New in MS 7.0 update 2.) Control whether or not to show per-recipient "spare" LDAP attributes.

Must be superuser or the MTA user in order to display -spares output.

71.34.4.45 -statistics, -nostatistics (default)

The -statistics switch can be used to tell the MTA to output the direct LDAP lookup cache statistics for the rewriting performed; statistics for the domain cache, reverse cache, and alias cache will be displayed. -nostatistics is the default.

71.34.4.46 -srs=([domain=*domain*],[secrets=*secrets*], [maxage=*maxage*])

The -srs switch provides a way to override certain SRS-related MTA configuration options: The domain value overrides the srs_domain MTA option, the secrets value overrides the srs_secrets MTA option, and the maxage> value overrides the srs_maxage value.

Additionally, specification of -nosrs causes test -rewrite to act as it none of the srs_* options are set.

71.34.4.47 -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 IMTA_TABLE:imta.filter; as of MS 7.0, the MTA system filter file is located as *CONFIGROOT*/imta.filter. The -system_filter switch 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 switch will prevent the MTA system filter file (legacy configuration) from being read in when there is no compiled configuration. These switches have no effect when an XML configuration is in use.

71.34.4.48 -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.

71.34.4.49 -tlsused, -notlsused (default)

(New in 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).

71.34.4.50 -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
```

71.34.4.51 -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.

71.34.5 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)**, showing 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 IMMNORMAL NOSERVICEALL (2)
  channel flags #1       = SMTP_CRLF MX IDENTNONENUMERIC DEFAULT
  channel flags #2       = COPYSENDPOST COPYWARNPOST POSTHEADBODY HEADERINC NOEXPROUTE
  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 RECEIVEDFROM VALIDATELOCALNONE NOTURN
  defaulthost            = domain.com domain.com
  linelength             = 998
  addrsperfile           = 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 innosoft.com)
  header reverse address = dan@innosoft.com
  envelope forw address  = dan@innosoft.com  (route (TCP-DAEMON,TCP-DAEMON)) (host innosoft.com)
  envelope rev address   = dan@innosoft.com  (route (TCP-DAEMON,TCP-DAEMON)) (host innosoft.com)
  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* *NOTIFY-DELAYS*

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 set on the channel (in legacy configuration, those channel options set 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` switch, or defaulting to the postmaster address). New in MS 7.0u2, 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.

The example below shows typical output generated by `imsimta test -rewrite` in MS 8.0.

```
% imsimta test -rewrite dan@innosoft.com
  address channel        = tcp_local                    (1)
  forward channel        = tcp_local                    (2)
  channel description    =
  channel caption        =
  channel user filter    =
  dest channel filter    =
  source channel filter  =
  phase filter  =
  channel flags #0       = BIDIRECTIONAL MULTIPLE IMMNONURGENT NOSERVICEALL  (3)
  channel flags #1       = SMTP_CRLF AFFINITYLIST IDENTNONENUMERIC DEFAULTDKIMIGNORE
  channel flags #2       = NOSENDPOST NOWARNPOST POSTHEADBODY HEADERINC NOEXPROUTE
  channel flags #3       = LOGGING NORESTRICTED RETAINSECURITYMULTIPARTS
  channel flags #4       = UTF8NEGOTIATE HEADERKEEPORDER NOHEADERREAD RULES
  channel flags #5       = TRUNCATESMTPLONGLINES
  defaulthost            = domain.com domain.com
  linelength             = 998
  addrsperfile           = 99
  channel env addr type  = SOURCEROUTE
  channel hdr addr type  = SOURCEROUTE
```

```
  channel official host  = tcp-daemon                        (4)
  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 daemon name    = outgate.domain.com
  channel user name      =
  urgentnotices          = 1 2 4 7
  normalnotices          = 1 2 4 7
  nonurgentnotices       = 1 2 4 7
  local behavior flags   = %x0
  expandchannel          =
  notificationchannel    =
  dispositionchannel     =
  backward channel       = tcp_local                         (5)
  unique identifier      = dan@innosoft.com
  header forward address = dan@innosoft.com  (route (TCP-DAEMON,TCP-DAEMON)) (host innosoft.com)
  header reverse address = dan@innosoft.com
  envelope forw address  = dan@innosoft.com  (route (TCP-DAEMON,TCP-DAEMON)) (host innosoft.com)
  envelope rev address   = dan@innosoft.com  (route (TCP-DAEMON,TCP-DAEMON)) (host innosoft.com)
  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* *NOTIFY-DELAYS* 4

Submitted notifications list:                              (7)

#
```

1. The channel to which, after performing any address reversal, the address matches an an envelope From address.

2. The channel which, rewriting as an envelope To address, the address matches.

3. The channel options set for the channel indicated in (2).

4. The channel's official host name as specified by the `official_host_name` option (Unified Configuration) or the second line of the channel block (legacy configuration) for the channel indicated in (2).

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` switch, or defaulting to the postmaster address).

71.34.6 Error messages

Usually errors reported by `imsimta test -rewrite` are not actually errors regarding `imsimta test -rewrite` in particular, but rather are the utility warning of an underlying configuration problem. For instance, "`Error in mm_init:` ..." sorts of errors are typically configuration errors (whether errors of syntax, of access, of inconsistent/incompatible settings, *etc.*).

`Address list error -- unknown host or domain:`

The above error indicates that the domain name in the specified address did not rewrite to any MTA channel. Check that the domain name was correctly spelled. If the domain name was correct and was a locally hosted domain, then most likely it is not correctly provisioned as a domain in LDAP; see the *Schema Reference*. If the domain name was a correctly spelled external domain name, then most likely you need a new (or changed) rewrite rule in the MTA configuration to handle that domain name, or an updated `tlds.txt` file; see TLD comparison rewrites.

`Unknown group identifier ... found on channel ...`

Prior to MS 7.0, any word on a channel not recognized as a channel option was interpreted as a group identifier. So prior to MS 7.0, the above error was possible, meaning that you (the executor of the `imsimta test -rewrite` command) do not have the specified group identifier. Check that the word shown is truly intended to be present as a group identifier, rather than simply being a misspelled channel option.

71.35 test -time utility

Test date-time strings.

71.35.1 Syntax

```
imsimta test -time input-string
```

Table 71.35 imsimta test -time Command Switches

Switch	Default
`-iso8601`	*None*
`-rfc822`	*None*
`-periodic`	`-noperiodic`
`-stamp`	*None*

71.35.2 Parameters

71.35.2.1 *string*

String to test for validity (and meaning) as a date-time.

71.35.3 Description

Test whether a string is a valid date-time string. See also the `imsimta test -zone` utility.

71.35.4 Switches

71.35.4.1 `-iso8601`, `-rfc822`, `-stamp`

The `-iso8601`, `-rfc822`, and `-stamp` qualifiers specify whether the input string is to be checked as a possible ISO 8601 time, an RFC 822 date-time, or as a "time stamp" string (such as from an MTA transaction log file entry), respectively. These qualifiers are mutually exclusive; one must be specified to obtain output. `-stamp` expects a 23 (or more) character long input, of the general form

DD-MMM-YYYY HH:MM:SS.cc

or alternatively

DD-MMM-YYYY:HH:MM:SS.cc

though as the *.cc* centiseconds are ignored, the output is determined only by the portion of the form:

DD-MMM-YYYY HH:MM:SS

or alternatively

DD-MMM-YYYY:*HH*:*MM*:*SS*

71.35.4.2 -periodic, -noperiodic

-periodic and -noperiodic (the default) switches are modifiers available with the -iso8601 qualifier, indicating whether ISO 8601 format is being tested, or ISO 8601 P format is being tested.

71.35.5 Examples

This example shows conversion of a time stamp (such as from an MTA transaction log entry) into other date-time formats.

```
# imsimta test -time -stamp "10-MAR-2013 12:13:14.55"
Result: 1362942794 Sun Mar 10 12:13:14 2013
After conversion to system time: "Sun, 10 Mar 2013 12:13:14 -0700 (PDT)"
After conversion back to time_t: 1362942794 Sun Mar 10 12:13:14 2013
```

This example shows checking the validity of an ISO 8601 format time.

```
# imsimta test -time -iso8601 "2014-06-01T020304.05"
Result: "Sun, 01 Jun 2014 02:03:04 -0700 (PDT)"
```

71.36 test -translation utility

Test charset translation.

71.36.1 Syntax

```
imsimta test -translation [input-string]
```

Table 71.36 imsimta test -translation Command Switches

Switch	Default
`-image_file=`*`filename`*	`-image_file=IMTA_TABLE:advanced/charset_data`
`-source=`*`charset-name`*	
`-destination=`*`charset-name`*	
`-input=`*`filename`*	
`-output=`*`filename`*	
`-strip_accent`	`-nostrip_accent`
`-call_reset`	`-nocall_reset`
`-generate`	`-nogenerate`
`-mnemonic_input`	`-nomnemonic_input`
`-utf8_input`	`-noutf8_input`

71.36.1.1 Prompts

Table 71.37 imsimta test -translate Prompts

Prompt	Value
`Destination character set:`	*`charset-name`*
`Source character set:`	*`charset-name`*
`Input string:`	*`string`*

71.36.2 Parameters

71.36.2.1 *input-string*

Input string to try translating to an alternate charset.

71.36.3 Description

The `imsimta test -translation` utility allows testing charset translation. Note that this is not *language* translation; rather, it is conversion of the representation of characters, from one charset representation, into another charset representation.

71.36.4 Switches

71.36.4.1 -image_file=*filename*

Specify the compiled charset data image file. By default, the image file used is IMTA_TABLE:advanced/charset_data (prior to MS 7.0, the file named by the imta_charset_data MTA Tailor option).

71.36.4.2 -source=*charset-name*

Specify a source charset: the charset used in the input string.

71.36.4.3 -destination=*charset-name*

Specify a destination charset: the charset to use in the output string.

71.36.4.4 -input=*filename*

Read input from the specified file.

71.36.4.5 -output=*filename*

Direct output to the specified file.

71.36.4.6 -strip_accent, -nostrip_accent (default)

When translating from one charset to another charset that does not have the actual character, but does have the character sans accent, specifying -strip_accent causes the accentless character to be output; effectively it "strips" accents. So for instance, -strip_accent may be useful when desiring to downgrade a charset such as ISO-8859-1 down to US-ASCII.

71.36.4.7 -call_reset, -nocall_reset (default)

Specifies whether or not the state of the charset converter is reset prior to each call.

71.36.4.8 -generate, -nogenerate (default)

Specifying -generate tells the utility to generate and output the entire charset table for converting from UTF-8 to the specified output charset. Specifying -generate implies -source=UTF-8. (Thus note that in particular, -generate and -source=*charset-name* cannot both be specified; and while some input file or string can be specified, it will be ignored.)

71.36.4.9 -mnemonic_input, -nomnemonic_input (default)

Specifies whether or not mnemonics can be used to enter non-US-ASCII characters.

71.36.4.10 -utf8_input, -noutf8_input (default)

Specifies whether or not escape sequences can be used to enter UTF-8 characters.

71.36.5 Examples

```
# imsimta test -trans -source=utf-8 -input=utf8.txt -dest=koi8-r -output=koi8r.txt
```

The above example shows a command line for translating the source file `utf8.txt` which is in the UTF-8 charset, into the destination file `koi8r.txt` which is in the KOI8-R charset.

```
# imsimta test -trans -source=koi8-r -dest=us-ascii -input=koi8r.txt -output=mnemonicized.txt
# cat mnemonicized.txt
U=V=E=L=I=C%I=T=%" P=R=O=D=A=Z%I=
```

"Converting" text to US-ASCII, when the original text contained characters not present in the US-ASCII charset, results in outputting the mnemonics (as defined in `charnames.txt`) for the "missing" characters. The above example shows outputting the character mnemonics corresponding to two (Russian language) words, originally represented in the KOI8-R charset; in this case, the output consists of mnemonics for a number of Cyrillic characters.

71.37 test -zone utility

Test a time zone string.

71.37.1 Syntax

```
imsimta test -zone string
```

71.37.2 Parameters

71.37.2.1 *string*

String to test for validity (and meaning) as a time zone name.

71.37.3 Description

Test whether a string is a recognized time zone name. See also the `imsimta test -time` utility.

71.37.4 Examples

This example shows a sample query on a recognized time zone name string. The offset shown is from GMT.

```
# imsimta test -zone PDT
Zone PDT decodes to -7 hours, 0 minutes.
```

This example shows testing a string that is not recognized by the MTA as a valid time zone name.

```
# imsimta test -zone bogus
Zone decode failed.
```

71.38 version utility

Print MTA version number.

71.38.1 Syntax

```
imsimta version
```

71.38.2 Parameters

None.

71.38.3 Description

`imsimta version` prints out the MTA version number, and displays the system's name, operating system release number and version, and hardware type.

As of Messaging Server 8.0 this utility also displays whether or not a compiled configuration is being used.

71.38.4 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.

71.39 view utility

Display the contents of the specified "version" of an MTA log file.

71.39.1 Syntax

```
imsimta view file-pattern
```

Table 71.38 imsimta view Command Switches

Switch	Default
`-f=offset-from-first`	*None*
`-l=offset-from-last`	*None*

71.39.1.1 Restrictions

Must have read access to the requested file.

71.39.2 Parameters

71.39.2.1 *file-pattern*

A file name pattern for which MTA log file to display. A complete file path may be specified. Or if merely a file name (sans unique id) is specified, the utility will look in the MTA log directory, `imta_log`.

71.39.3 Description

The `imsimta view` utility may be used to display a specified "version" of an MTA log file. MTA log files have a *-uniqueid* appended to the file name to allow for the creation of multiple "versions" of the log file; on UNIX, the *-uniqueid* is appended to the very end of the file name (the end of the file extension). The `imsimta view` utility understands these unique ids and can display the contents of the particular file corresponding to the requested "version" of the file.

The default, if no offset switch is specified, is to display the most recent "version" of the file.

71.39.4 Switches

71.39.4.1 `-f=`*offset-from-first*

This switch is used to specify displaying the *n*th "version" of the file (starting counting from 0). For instance, to display the earliest (oldest) "version" of the file, specify `-f=0`

71.39.4.2 `-l=`*offset-from-last*

This switch is used to specify displaying the *n*th from the last "version" of the file (starting decrementing from 0 as the most recent version). For instance, to display the most recent (newest) "version" of the file, specify `-l=0`

71.39.5 Examples

```
# imsimta view -l=0 job_controller.log
fat_main: watch_connect failed: Connection refused
```

The above example shows displaying the most recent Job Controller log file, and finding logged therein that the Watcher though enabled (`watcher.enable` in Unified Configuration, or `local.watcher.enable` in legacy configuration) was apparently not running (which is what the error message shown in that log file indicates).

```
# imsimta view -f=0 tcp_intranet_master.log
13:00:16.79: Checking for TLS usage
13:00:16.79: Initializing TLS library
13:00:16.98: SMTP options read and set
13:00:16.98: Debug output enabled, system mail.domain.com, process 6597.1, SMTP client version V8.0 compiled Aug 11 2014 11:
13:00:16.98: Oracle Communications Messaging Server shared library version 8.0.0.0.0 linked 11:42:42, Aug  7 2014
13:00:16.98:      max concurrent threads 10, jobs/thread 10
13:00:16.98: ( 2)  tcp_intranet running
13:00:16.98: ( 2)  tcp_intranet -- delivering /opt/sun/comms/messaging64/data/queue/tcp_intranet/004/ZZi0Z6h0Ykfc0.00
13:00:30.29: waiting for 1 threads to exit, or 5400 seconds
13:00:32.31: ( 2)  tcp_intranet finished. did 1 messages
13:00:32.31: ( 2) No more hosts, ending connections
13:00:32.31: All message processing completed.
# imsimta view -f=1 tcp_intranet_master.log
13:00:16.98: Initializing message dequeue via quc_rinit, file "/opt/sun/comms/messaging64/data/queue/tcp_intranet/004/ZZi0Z6
13:00:16.98: Reading first To: address
13:00:16.98: log_queue_entry called.
13:00:16.98:   Originator address john.doe@domain.com
13:00:16.98:   Recipient address jane.doe@domain.com
13:00:16.98:   Original recipient address rfc822;jane.doe@domain.com
13:00:16.98: Forced routing to host1.domain.com
13:00:16.98: Setting up connection to "host1.domain.com", initial mailbox "jane.doe"
13:00:16.98: No connection currently open
13:00:16.98: Opening new connection for host1.domain.com
...etc...
```

The above example shows displaying two `tcp_intranet_master.log-*` files containing `master_debug` output, the (currently) oldest (which in this case is "controller" thread output), and the (currently) next-to-oldest (which in this case is output from a delivery thread, processing a particular message file).

Part VIII Additional components

Messaging Server includes a Personal Addressbook facility, SNMP support, and support for an Event Notification Service.

Chapter 72 PAB options

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.

72.1 `enable` Option Under pab

The `enable` PAB option enables or disables the Personal Address Book (PAB) feature.

72.2 `defaulthostindex` Option

The `defaulthostindex` PAB option specifies the index of the default host.

72.3 `active` Option

The `active` PAB option should be set to 1 if PAB host is active, 0 otherwise.

72.4 `alwaysusedefaulthost` Option

The `alwaysusedefaulthost` PAB option enables one PAB server to be used (overriding hostname in PAB URIs).

72.5 `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.

72.6 `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).

72.7 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).

72.8 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).

72.9 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).

72.10 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).

72.11 ldapusessl Option

The ldapusessl PAB option specifies whether to use SSL to connect to the PAB Directory Server.

72.12 maxnumberofentries Option

The maxnumberofentries PAB option specifies the maximum number of entries a single PAB can store.

72.13 migrate415 Option

The migrate415 PAB option enables PAB migration when set to 1.

72.14 numberofhosts Option

The numberofhosts PAB option specifies the number of PAB servers (up to a maximum of 16).

Chapter 73 SNMP options

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.

Note that while Messaging Server's SNMP support normally runs as a SNMP subagent, receiving SNMP requests via the platform's SNMP master agent, `snmpd`, Messaging Server's SNMP subagent can be run in a different mode, as an SNMP agent itself, independent of `snmpd`; see the `standalone`, `listenaddr`, and `port` SNMP options.

See also the logfile options set as `snmp.logfile.*`.

73.1 `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.

73.2 `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; (this is the default). 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.

73.3 `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`. The default is `161`.

This option applies only to the Net-SNMP based SNMP subagent, but *not* to the legacy SNMP subagent for Solaris 9 and earlier.

73.4 cachettl Option Under snmp

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.

73.5 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.

73.6 directoryscan Option

The directoryscan SNMP option selects whether or not the subagent performs scans of the MTA's 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; see MIB variables served. 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.

73.7 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.

73.8 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.

73.9 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.

73.10 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 74 ENS options

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.

74.1 `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).

74.2 `port` Option Under ens

The `port` ENS option specifies the TCP port the ENS server will listen on. The default is `7997`. 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.

74.3 `enablesslport` Option Under ens

The `enablesslport` ENS option sets whether or not ENS over SSL service is started. Note that the `ens.sslport` option controls what port *is* the ENS+SSL port.

74.4 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.enablesslport option must be set.

74.5 secret Option Under ens

The secret ENS option specifies the secret used to authenticate ENS clients with the server.

74.6 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 the Unified Configuration logfile.loglevel option (logfile.*.loglevel in legacy configuration). But note that the value "debug" generates lots of data and is not recommended.

74.7 domainallowed Option Under ens

The domainallowed ENS option specifies access filters specifying which domains and/or IP addresses are allowed ENS access.

74.8 domainnotallowed Option Under ens

The domainnotallowed ENS option specifies access filters specifying which domains and/or IP addresses are not allowed ENS access.

74.9 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).

74.10 mustauthenticate Option

The mustauthenticate option under ens, ens.mustauthenticate, specifies whether the ENS server requires authentication.

Chapter 75 eval_ldapd options

The Evaluation LDAP server (eval_ldapd) supports options controlling access via access filters.

75.1 `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.

75.2 `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.

Table A.1 *Protocols*

Reference	Status	Feature and Information
RFC 3501	Proposed standard	*IMAP* (port 143) (Usage: Server Client) (Older specs: RFC2060) March 2003
RFC 2180	Informational	*IMAP4 Multi-Accessed Mailbox Practice* (port 143) (Usage: Server Client) July 1997
RFC 2061	Informational	*IMAP4 Compatibility with IMAP2bis* (port 143) (Usage: Server Client) (Older specs: RFC1730) December 1996
RFC 2062	Informational	*Internet Message Access Protocol - Obsolete Syntax* (port 143) (Usage: Server Client) December 1996
RFC 2683	Informational	*IMAP4 Implementation Recommendations* (port 143) (Usage: Server Client) September 1999
RFC 1730	Proposed standard	*IMAP4* (Usage: Server) (Newer specs: RFC2060 RFC2061) December 1994
RFC 3501 section 7.2.1	Proposed standard	*IMAP4rev1* Caveat: Cassandra store and ISS search are not complaint with the IMAP specification. (Usage: Server Client) (Older specs: RFC2060) March 2003

RFC 3501 section 6.2.1	Proposed standard	*IMAP STARTTLS* (Usage: Server Off-by-default) (Older specs: RFC2060) March 2003
RFC 8446		*TLS 1.3 as used by IMAP STARTTLS* (Usage: Server Off-by-default)
RFC 3501 section 6.2.3	Proposed standard	*IMAP LOGINDISABLED* (Usage: Server Off-by-default) (Older specs: RFC2060) March 2003
RFC 4422	Proposed standard	*AUTHENTICATE (SASL)* Caveat: SASLprep is not implemented. (Usage: Server Client) (Older specs: RFC2222) June 2006
RFC 3501 section 6.2.2	Proposed standard	*AUTHENTICATE (SASL)* Caveat: SASLprep is not implemented. (Usage: Server Client) (Older specs: RFC2060) March 2003
RFC 4616	Proposed standard	*IMAP AUTH=PLAIN* Caveat: SASLprep is not implemented. (Usage: Server Client) August 2006
RFC 4422 page 29	Proposed standard	*IMAP AUTH=EXTERNAL* (Usage: Server Off-by-default) (Older specs: RFC2222) June 2006
RFC 2245	Proposed standard	*IMAP AUTH=ANONYMOUS* (Usage: Server Off-by-default) (Newer specs: RFC4505) November 1997
RFC 2195	Proposed standard	*IMAP AUTH=CRAM-MD5* (Usage: Server Off-by-default) (Older specs: RFC2095) September 1997
RFC 2086	Proposed standard	*IMAP ACL* Caveat: Support for the 'p' right only works for user 'anyone' presently. (Usage: Server) (Newer specs: RFC4314) January 1997
RFC 2087	Proposed standard	*IMAP QUOTA* (Usage: Server) January 1997
RFC 2088	Proposed standard	*IMAP LITERAL+* (Usage: Server) January 1997
RFC 2342	Proposed standard	*IMAP NAMESPACE* (Usage: Server) May 1998

RFC 2359	Proposed standard	*IMAP UIDPLUS* (Usage: Server) (Newer specs: RFC4315) June 1998
RFC 3348	Informational	*IMAP CHILDREN* (Usage: Server) July 2002
RFC 3516	Proposed standard	*IMAP BINARY* (Usage: Server) April 2003
RFC 3691	Proposed standard	*IMAP UNSELECT* Added in release: 6.3 (Usage: Server) February 2004
RFC 2177	Proposed standard	*IMAP IDLE* Added in release: 6.3 (Usage: Server Off-by-default) (Enhancement Request: 12037435) June 1997
RFC 4959	Proposed standard	*IMAP SASL-IR* Added in release: 7.0 (Usage: Server) September 2007
RFC 4469	Proposed standard	*IMAP CATENATE* Added in release: 7.0 (Usage: Server) April 2006
RFC 5161	Proposed standard	*IMAP ENABLE* Added in release: 7.0 (Usage: Server) March 2008
RFC 4731	Proposed standard	*IMAP ESEARCH* Added in release: 7.0 (Usage: Server) November 2006
RFC 4551	Proposed standard	*IMAP CONDSTORE* Added in release: 7.0 (Usage: Server) (Newer specs: RFC7162) June 2006
RFC 5092	Proposed standard	*IMAP URL* 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	*IMAP URLAUTH* Added in release: 7.0 (Usage: Server Client) May 2006
RFC 5162	Proposed standard	*IMAP QRESYNC* Added in release: 7.0 (Usage: Server) (Newer specs: RFC7162) March 2008
RFC 5032	Proposed standard	*IMAP WITHIN* Added in release: 7.0 (Usage: Server) September 2007

RFC 2221	Proposed standard	*IMAP Login Referrals* Details: Used to detect user in transit via rehostuser utility. Added in release: 7.0 (Usage: Server Client) October 1997
RFC 5257	Experimental	*IMAP ANNOTATE* Added in release: 7.0 (Usage: Server) June 2008
RFC 5267	Proposed standard	*IMAP CONTEXT* Added in release: 7.0 (Usage: Server) July 2008
RFC 5256	Proposed standard	*IMAP THREAD* 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	*IMAP SORT* 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	*IMAP SORT=DISPLAY* 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	*IMAP LANGUAGE* (Usage: Server Client) June 2008
RFC 5464	Proposed standard	*IMAP METADATA* Added in release: 7.0.5 (Usage: Server) February 2009
RFC 2971	Proposed standard	*IMAP ID* Added in release: 7 Update 4 (Usage: Server) (Enhancement Request: 12139651) October 2000
RFC 5530	Proposed standard	*IMAP Response Codes* Added in release: 7.0.5 (Usage: Server) May 2009
RFC 4314	Proposed standard	*IMAP ACL Extension* (Usage: Server) (Older specs: RFC2086) December 2005

RFC 5182	Proposed standard	*IMAP Referencing the Last SEARCH Result* Added in release: 7 Update 2 (Usage: Server) March 2008
RFC 5819	Proposed standard	*IMAP4 Extension for Returning STATUS Information in Extended LIST* Added in release: 8.0 (Usage: Server) (Enhancement Request: 16773851) March 2010
RFC 6154	Proposed standard	*IMAP LIST Extension for Special-Use Mailboxes* Added in release: 8.0 (Usage: Server) (Enhancement Request: 16773916) March 2011
RFC 7377	Proposed standard	*IMAP MULTISEARCH* Added in release: 8.0 (Usage: Server) (Older specs: RFC6237) (Enhancement Request: 17596568) October 2014
RFC 2221	Proposed standard	*IMAP LOGIN-REFERRALS* Added in release: 8.0.1 (Usage: Server Client) (Enhancement Request: 21092120) October 1997
RFC 6855	Proposed standard	*IMAP UTF8=ACCEPT* Caveat: The implementation is more permissive about using/allowing UTF-8 in the protocol than the specification. Added in release: 8.1 (Usage: Server) (Older specs: RFC5738) March 2013
RFC 5550	Proposed standard	*IMAP URL-PARTIAL* Added in release: 8.0.2.3 (Usage: Server) (Older specs: RFC4550) August 2009
RFC 8438		*IMAP STATUS=SIZE* Added in release: 8.0.2.3 (Usage: Server)
RFC 8437		*UNAUTHENTICATE* Added in release: future (Usage: Server)
RFC 8514		*SAVEDATE* Added in release: future (Usage: Server)
RFC 1939	Internet standard	*POP3* (port 110) (Usage: Server Client) (Older specs: RFC1725) May 1996

RFC 1957	Informational	*Some Observations on Implementations of the Post Office Protocol (POP3)* (port 110) (Usage: Server Client) June 1996
RFC 1939 section 7	Internet standard	*POP TOP* (Usage: Server) (Older specs: RFC1725) May 1996
RFC 1939 page 12	Internet standard	*POP UIDL* (Usage: Server Client) (Older specs: RFC1725) May 1996
RFC 1939 page 13	Internet standard	*POP USER / PASS* (Usage: Server Client) (Older specs: RFC1725) May 1996
RFC 1939 page 15	Internet standard	*APOP* (Usage: Server Off-by-default) (Older specs: RFC1725) May 1996
RFC 2449 section 6.3	Proposed standard	*POP SASL* (Usage: Server Client) November 1998
RFC 2449	Proposed standard	*POP CAPA* Caveat: We don't implement EXPIRE or LOGIN-DELAY. (Usage: Server) November 1998
RFC 2449 section 6.6	Proposed standard	*POP PIPELINING* (Usage: Server) November 1998
RFC 2449 section 6.4	Proposed standard	*POP RESP-CODES* (Usage: Server) November 1998
RFC 2449 section 6.9	Proposed standard	*POP IMPLEMENTATION* (Usage: Server) November 1998
RFC 2595 section 4	Proposed standard	*POP STLS* (Usage: Server Off-by-default) June 1999
RFC 8446		*TLS 1.3 as used by POP STLS* (Usage: Server Off-by-default)
RFC 3206	Proposed standard	*POP AUTH-RESP-CODES* (Usage: Server) February 2002
RFC 5321	Draft standard	*SMTP Specification* (port 25) (Usage: Server Client) (Older specs: RFC2821) October 2008
RFC 821	Internet standard	*SMTP Legacy Specification* (port 25) (Usage: Server Client) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 920	Unknown	*Domain requirements* (port 25) (Usage: Server Client) October 1984

RFC 974	Historic	*SMTP Mail Routing and DNS* (port 25) (Usage: Server Client) (Newer specs: RFC2821) January 1986
RFC 1123 section 5	Internet standard	*SMTP Host Requirements* (port 25) (Usage: Server Client) October 1989
RFC 2505	Best current practice	*Anti-Spam Recommendations for SMTP MTAs* (port 25) (Usage: Server Client) February 1999
RFC 3848	Draft standard	*ESMTP and LMTP Transmission Types Registration* (port 25) (Usage: Server Client) July 2004
RFC 1428	Informational	*Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME* (port 25) (Usage: Server Client) February 1993
RFC 6152	Internet standard	*SMTP 8BITMIME* (Usage: Server Client) (Older specs: RFC1652) March 2011
RFC 2920	Internet standard	*SMTP PIPELINING* (Usage: Server Client) (Older specs: RFC2197) September 2000
RFC 3030	Proposed standard	*SMTP CHUNKING* 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	*SMTP DSN* (Usage: Server Client) (Older specs: RFC1891) January 2003
RFC 2034	Proposed standard	*SMTP ENHANCEDSTATUSCODES* (Usage: Server Client) October 1996
RFC 3463	Draft standard	*Enhanced Mail System Status Codes* (Usage: Server Client) (Older specs: RFC1893) January 2003
RFC 2821 section 4.1.1.7	Proposed standard	*SMTP EXPN* (Usage: Server) (Older specs: RFC0821 RFC0974 RFC1869) (Newer specs: RFC5321) April 2001
RFC 2821 section 4.1.1.8	Proposed standard	*SMTP HELP* (Usage: Server) (Older specs: RFC0821

		RFC0974 RFC1869) (Newer specs: RFC5321) April 2001
RFC 821 section 3.4	Internet standard	*SMTP SAML* (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.4	Internet standard	*SMTP SEND* (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.4	Internet standard	*SMTP SOML* (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 821 section 3.8	Internet standard	*SMTP TURN* (Usage: Server Off-by-default) (Older specs: RFC0788) (Newer specs: RFC2821) August 1982
RFC 3207	Proposed standard	*SMTP STARTTLS* (Usage: Server Client Off-by-default) (Older specs: RFC2487) February 2002
RFC 8446		*TLS 1.3 as used by SMTP STARTTLS* (Usage: Server Client Off-by-default)
RFC 4954	Proposed standard	*SMTP AUTH* 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	*SMTP AUTH* 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	*SMTP ETRN* (Usage: Server) August 1996

RFC 1870	Internet standard	*SMTP SIZE* (Usage: Server) (Older specs: RFC1653) November 1995
RFC 1846	Experimental	*SMTP 521 Reply Code* (Usage: Client) September 1995
RFC 7505	Proposed standard	*Null MX No Service Resource Record* Details: Used to provide more timely reporting of SMTP errors Added in release: 8.0 (Usage: Client) June 2015
RFC 7372	Proposed standard	*Email Authentication Status Codes* Details: Used to provide more accurate reporting of SMTP DNS errors Added in release: 8.0 (Usage: Client) September 2014
RFC 3865	Proposed standard	*SMTP NO-SOLICITING* (Usage: Server Off-by-default) September 2004
SMTP AUTH=LOGIN	Widely Used or De Facto	*SMTP AUTH=LOGIN* Details: Undocumented pre-standard subset of AUTH PLAIN but with interoperability problems. (Usage: Server)
RFC 4408	Experimental	*Sender Permitted From* (Usage: Client) (Newer specs: RFC7208) April 2006
RFC 2852	Proposed standard	*SMTP Deliver By* 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	*SMTP ALTRECIP* Details: Delivery to alternate recipient on error Added in release: 8.0.1 (Usage: Client)
RFC 6710	Proposed standard	*SMTP Priority* Details: Envelope specification of message processing priority Added in release: 8.0 (Usage: Server Client) August 2012
RFC 6710	Proposed standard	*SMTP Priority Tunneling* Details: Tunneling of SMTP Message Transfer Priorities Added in release: 8.0.1 (Usage: Server Client) August 2012
RFC 6729	Proposed standard	*Received state* Details: Received: field indicator of the

		state attached to the message Added in release: 8.0 (Usage: Server) September 2012
RFC 7293	Proposed standard	*Require Recipient Valid Since* Details: Mailbox ownership change detection Added in release: 8.0 (Usage: Server Client) July 2014
RFC 6409	Internet standard	*Message Submission* Details: Superset of SMTP adding AUTH by default. (port 587) (Usage: Server Client) (Older specs: RFC4409) November 2011
RFC 4954	Proposed standard	*Submission AUTH* 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: RFC2554) July 2007
RFC 4422	Proposed standard	*Submission AUTH* 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	*Submission AUTH PLAIN* Caveat: SASLprep is not implemented. (Usage: Server Client) August 2006
RFC 4422 page 29	Proposed standard	*Submission AUTH=EXTERNAL* (Usage: Server Client Off-by-default) (Older specs: RFC2222) June 2006
Submission AUTH=LOGIN	Widely Used or De Facto	*Submission AUTH=LOGIN* Details: Undocumented, subset of AUTH PLAIN functionality. (Usage: Server Client)
RFC 4468	Proposed standard	*Submission BURL* Added in release: 7.0 (Usage: Server) May 2006

RFC 4865	Proposed standard	*Submission Future Release* Added in release: 7.0 (Usage: Server) May 2007
RFC 3887	Proposed standard	*MTQP Specification* Details: Message Tracking Added in release: 8.0 (port 1038) (Usage: Server Client) September 2004
RFC 3886	Proposed standard	*Tracking response format* Details: Message Tracking Added in release: 8.0 (port 1038) (Usage: Server Client) September 2004
RFC 3885	Proposed standard	*SMTP MSGTRK extension* Details: Message Tracking Added in release: 8.0 (port 1038) (Usage: Server Client) September 2004
RFC 3888	Informational	*Tracking model and requirements* Details: Message Tracking Added in release: 8.0 (port 1038) (Usage: Server Client) September 2004
RFC 3458	Proposed standard	*Message Context for Internet Mail* Details: Gateway for pager/cell phone connectivity for SMS. (Usage: Server Client Off-by-default) January 2003
RFC 2251	Proposed standard	*LDAPv3* (port 389) (Usage: Client) (Newer specs: RFC4510 RFC4511 RFC4513 RFC4512) December 1997
RFC 8314		*IMAPS* Details: standard IMAP SSL port (port 993) (Usage: Server Client Off-by-default)
RFC 8314		*POP3S* Details: standard POP3 SSL port (port 995) (Usage: Server Client Off-by-default)
RFC 8314		*SUBMISSIONS* Details: standard mail submission SSL port (port 465) (Usage: Server Client Off-by-default)
RFC 1034	Internet standard	*Domain names - concepts and facilities* (port 53) (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035	Internet standard	*Domain names - implementation and specification* (port 53)

		(Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.4.1	Internet standard	*DNS A* (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 3596	Draft standard	*DNS AAAA* Details: Support can be configured on (Usage: Client) (Older specs: RFC3152 RFC1886) October 2003
RFC 1035 section 3.3.1	Internet standard	*DNS CNAME* (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.3.9	Internet standard	*DNS MX* (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 934	Unknown	*DNS MX* (Usage: Client) January 1985
RFC 1035 section 3.3.12	Internet standard	*DNS PTR* (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 1035 section 3.3.14	Internet standard	*DNS TXT* Details: For RBL and SPF. (Usage: Client) (Older specs: RFC0973 RFC0882 RFC0883) November 1987
RFC 3507	Informational	*ICAP* Details: For Norton AV. (port 1344) (Usage: Client Off-by-default) April 2003
RFC 1928	Proposed standard	*SOCKS 5* (port 1080) (Usage: Client Off-by-default) March 1996
RFC 2788	Proposed standard	*SNMP Network Services MIB* Details: Provided via NetSNMP. (port 161) (Usage: Server Off-by-default) (Older specs: RFC2248) March 2000
RFC 2789	Proposed standard	*SNMP Mail MIB* Details: Provided via NetSNMP. (port 161) (Usage: Server Off-by-default) (Older specs: RFC2249 RFC1566) March 2000
RFC 2033	Informational	*LMTP* 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

Table A.2 ***Data and File Formats, Schema***

Reference	Status	Feature and Information
RFC 822	Internet standard	*Legacy Internet Message Format* (Usage: Generate Accept Process) (Older specs: RFC0733) (Newer specs: RFC2822) August 1982
RFC 5322	Draft standard	*Internet Message Format* (Usage: Generate Accept Process) (Older specs: RFC2822) October 2008
RFC 6532	Proposed standard	*Internationalized Email Headers* (Usage: Generate Accept Process) (Older specs: RFC5335) February 2012
RFC 2045	Draft standard	*Multi-purpose Internet Mail Extensions (MIME)* (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2046	Draft standard	*Multi-purpose Internet Mail Extensions (MIME)* (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2049	Draft standard	*Multi-purpose Internet Mail Extensions (MIME)* (Usage: Generate Accept Process) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2047	Draft standard	*MIME International Headers* (Usage: Generate Accept Process Normalize) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 2231	Proposed standard	*MIME International Parameters* (Usage: Accept Process)

		(Older specs: RFC2184) November 1997
RFC 2183	Proposed standard	*MIME Content-Disposition* (Usage: Generate Accept Process) (Older specs: RFC1806) August 1997
RFC 1864	Draft standard	*Content-MD5* (Usage: Process Validate) (Older specs: RFC1544) October 1995
RFC 3458	Proposed standard	*Message Context for Internet Mail* Details: For per-context quotas. Added in release: 6.3 (Usage: Generate Accept) January 2003
RFC 1740	Proposed standard	*Macintosh MIME Format* (Usage: MTA Translate Off-by-default) December 1994
RFC 1741	Informational	*Macintosh Binhex Encoding* (Usage: MTA Translate Off-by-default) December 1994
RFC 2045 section 6.7	Draft standard	*MIME base-64 and quoted-printable encodings* (Usage: Generate Accept Process MTA Translate Validate Normalize) (Older specs: RFC1521 RFC1522 RFC1590) November 1996
RFC 1847	Proposed standard	*MIME multipart/signed* (Usage: Process) October 1995
RFC 2480	Proposed standard	*Gateways and MIME Security Multiparts* (Usage: Process) January 1999
RFC 3280	Proposed standard	*X.509 Certificate Profile* 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	*MIME multipart/report* (Usage: Generate Process) (Older specs: RFC1892) (Newer specs: RFC6522) January 2003
RFC 3464	Draft standard	*Delivery Status Notifications (DSN)* (Usage: Generate Process) (Older specs: RFC1894) January 2003
RFC 3798	Draft standard	*Message Disposition Notifications (MDN)* (Usage: Generate Process Off-

		by-default) (Older specs: RFC2298) May 2004
RFC 2442	Informational	*Batch SMTP* (Usage: Generate Accept Off-by-default) November 1998
RFC 5228	Proposed standard	*Sieve (Mail Filtering Language)* (Usage: Accept) (Older specs: RFC3028) January 2008
RFC 5173	Proposed standard	*Sieve body* Details: Includes some storage and complexity limits. Added in release: 7 Update 3 (Usage: Accept Off-by-default) April 2008
RFC 3894	Proposed standard	*Sieve :copy parameter* (Usage: Accept) October 2004
RFC 5260	Proposed standard	*Sieve date and index* Added in release: 7 Update 4 (Usage: Accept) (Enhancement Request: 12183152) July 2008
RFC 7352	Proposed standard	*Sieve duplicate* Added in release: 8.0 (Usage: Accept) September 2014
RFC 5293	Proposed standard	*Sieve Editheader* Added in release: 6.3 (Usage: Accept) August 2008
RFC 5228	Proposed standard	*Sieve Encoded-character extension* Added in release: 7.0 (Usage: Accept) (Older specs: RFC3028) January 2008
RFC 6069	Experimental	*Sieve envelope-dsn, redirect-dsn* Details: Messaging Server Support for envelope DSN access. Added in release: 7 Update 2 (Usage: Accept) December 2010
RFC 5183	Proposed standard	*Sieve Environment* Added in release: 7 Update 1 (Usage: Accept) May 2008
RFC 5429	Proposed standard	*Sieve ereject* 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	*Sieve ihave* Added in release: 7.0 (Usage: Accept) March 2009
RFC 5232	Proposed standard	*Sieve imap4flags* Added in release: 6.3p1 (Usage: Accept) (Enhancement Request: 12108510) January 2008
RFC 5703	Proposed standard	*Sieve MIME extension* 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	*Sieve Notifications by email* Caveat: We only implement notify method email. Added in release: 6.2 (Usage: Accept Off-by-default) (Enhancement Request: 12070010)
RFC 5435	Proposed standard	*Sieve extension for notifications* Caveat: Only mailto: URLs are presently supported Added in release: 7.0.5 (Usage: Accept) January 2009
RFC 5436	Proposed standard	*Sieve notification mechanism: mailto* Added in release: 7.0.5 (Usage: Accept) January 2009
RFC 5231	Proposed standard	*Sieve Relational Tests* (Usage: Accept) (Older specs: RFC3431) January 2008
RFC 5233	Proposed standard	*Sieve subaddress* Added in release: 6.0 (Usage: Accept) (Older specs: RFC3598) January 2008
RFC 5235	Proposed standard	*Sieve Spamtest / Virustest* (Usage: Accept) (Older specs: RFC3685) January 2008
RFC 5230	Proposed standard	*Sieve vacation* (Usage: Accept) January 2008
RFC 6131	Proposed standard	*Sieve vacation seconds* Added in release: 7.0.5 (Usage: Accept) July 2011

RFC 5229	Proposed standard	*Sieve variables* Added in release: 6.3 (Usage: Accept) January 2008
RFC 952 section 1	Unknown	*Domain Name Syntax* (Usage: Accept Validate) (Older specs: RFC0810) October 1985
RFC 1123 section 2.1	Internet standard	*Domain Name Syntax* (Usage: Accept Validate) October 1989
RFC 2251	Proposed standard	*namingContexts of root DSE* 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	*LDAP Data Interchange Format (LDIF)* (Usage: Generate) June 2000
RFC 2254	Proposed standard	*LDAP Search Filter String Format* (Usage: Generate Accept Validate) (Older specs: RFC1960) (Newer specs: RFC4510 RFC4515) December 1997
RFC 2253	Proposed standard	*LDAP Distinguished Name String Format* (Usage: Generate MTA Translate Normalize) (Older specs: RFC1779) (Newer specs: RFC4510 RFC4514) December 1997
RFC 2255	Proposed standard	*LDAP URL Format* (Usage: Accept Validate) (Older specs: RFC1959) (Newer specs: RFC4510 RFC4516) December 1997
RFC 3280	Proposed standard	*PKIX* 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	*ASN.1* Caveat: Only the LDAP, PKIX and SNMP subsets are implemented. (Usage: Generate Accept Validate)
RFC 1738 section 3.10	Proposed standard	*File URL Format* (Usage: Accept) (Newer specs: RFC4248 RFC4266) December 1994
RFC 3339	Proposed standard	*Internet Date / Time Timestamps* 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	*Unix /var/mail mailbox file format* Details: Traditional 'From ' Unix mailbox format (multiple messages per file). Used by MTA native channel and migration tool. (Usage: Generate MTA Translate)
RFC 8259		*application/json* Details: Used for Elastic search, ISS, etc. (Usage: Generate Accept)

Table A.3 ***Charsets***

Reference	Status	Feature and Information
ANSI X3.4-1986	Other National/International	*US-ASCII* Details: ANSI X3.4-1986 (Usage: Accept MTA Translate Validate IMAP Search)
RFC 3629	Internet standard	*UTF-8* 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	*ISO 8859 (1-10,14,15)* 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	*ISO 8859 11* 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	*GB-2312* Details: Simplified Chinese, also CNS 11643-1986. (Usage: MTA Translate IMAP Search)
RFC 1842	Informational	*HZ-GB-2312* Details: Simplified Chinese. No MTA support. (Usage: IMAP Search) August 1995
Big5		*Big5* Details: Traditional Chinese (Usage: MTA Translate IMAP Search)
GB-18030	National/International	*GB-18030* Details: Extended Simplified Chinese with Unicode subset. (Usage: MTA Translate) (Enhancement Request: 12068067)
ISO/IEC 2022-1986	Other National/International	*ISO-2022* 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	*ISO-2022-JP* Details: Japanese, based on ISO 2022, JIS X 0201-1976, JIS X 0208-1990 (Usage: MTA Translate IMAP Search) June 1993
RFC 1554	Informational	*ISO-2022-JP-2* 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	*EUC-JP* Details: Japanese. (Usage: MTA Translate IMAP Search)
RFC 1557	Informational	*ISO-2022-KR* Details: Korean KSC 5601-1987 with ISO 2022 encoding (Usage: MTA Translate IMAP Search) December 1993
RFC 1557	Informational	*EUC-KR* Details: Korean KSC 5601-1987 with EUC encoding

		(Usage: MTA Translate IMAP Search) December 1993
RFC 1345	Informational	*ks_c_5601-1987* 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	*KOI8-R* Details: Russian 8-bit with US-ASCII subset. (Usage: MTA Translate IMAP Search) July 1993
Thai Industrial Standard 620-2533	Other National/International	*TIS-620* Details: TIS-620. IMAP SEARCH support new in 7u4 (Usage: MTA Translate IMAP Search)
Windows-1250	Vendor Non-standard, subject to change	*Windows-1250* 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	*Windows-1251* 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	*Windows-1252* Details: Windows variant of ISO-8859-1 (Latin) with extensions. (Usage: MTA Translate IMAP Search)
Windows-1253	Vendor Non-standard, subject to change	*Windows-1253* 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	*Windows-1254* Details: Windows variant of ISO-8859-9 (Turkish) with extensions. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
Windows-1255	Vendor Non-standard, subject to change	*Windows-1255* 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	*Windows-1256* Details: Windows incompatible variant of ISO-8859-6 (Arabic) with extensions. (Usage: MTA Translate IMAP Search)
Windows-1257	Vendor Non-standard, subject to change	*Windows-1257* 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	*Windows-1258* Details: Windows Vietnamese 8-bit. IMAP SEARCH support new in 7u3 (Usage: MTA Translate IMAP Search)
RFC 2152	Informational	*UTF-7* 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	*IMAP Modified UTF-7* (Usage: Accept MTA Translate Validate) (Older specs: RFC2060) March 2003

Table A.4 ***Library files***

Reference	Status	Feature and Information
RFC 1345	Informational	*charnames.txt* Details: Defines the mnemonic character names (based on RFC 1345) used in charsets.txt (Usage: Accept) June 1992
RFC 1345	Informational	*charsets.txt* Details: Character set definitions based on RFC 1345 (Usage: Accept) June 1992
ISO 3166 country codes	Other National/International	*countries.txt* Details: List of ISO 3166 country codes (Usage: Accept)

IANA top-level domain list	Other National/International	*tlds.txt* Details: Copy of http://data.iana.org/TLD/tlds-alpha-by-domain.txt (Usage: Accept)
RFC 3066	Best current practice	*languages.txt* 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 `channel` 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 necessarily 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 for instance the `ssladjustciphersuites` option. See Also SSL (Security Sockets Layer), TLS (Transport Layer Security).
Common name	In X.500 terminology, the common name or commonName or CN 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.

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 the `FROM_ACCESS` mapping table or recipient address `*_ACCESS` mapping tables (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, *etc.*, 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).
Domain alias	A domain alias is another, synonymous, name used for a domain. The MTA supports various forms and mechanisms for domain aliases, some simple and some complex. See in particular the `ldap_domain_attr_alias` MTA option (Schema 1), the `ldap_attr_domain2_schema2` MTA option (Schema 2), and the `domain_uplevel` MTA option. See Also Virtual domain.

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, *e.g.*, 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 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.

Group

In the email context, shorthand for mail group.
See Also Mail group.

GUI (Graphical User Interface)

A Graphical 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 scanningand 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

LDIF (LDAP Data Interchange Format)

LDIF is a standard plain text format for representing LDAP directory content; see RFC 2849 (LDIF).

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).

Mail group

A set of addresses (or aliases to expand) which receive email addressed to some one address: a "mail forwarder". This is distinguished from true mailing lists, as mailing lists have additional semantics. A mail group is intended, and normally should only be used, for a small number of closely related addresses/recipients -- such as a group of mail system administrators, or a group of family members. See Proper use of lists rather than groups for further discussion.
See Also Mailing list.

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.

Mailing list

A mailing list consists, at the most basic, of a mailing list address, a list owner address, and a set of member addresses to whom to post messages addressed to the list. Mailing lists optionally have many other configuration options -- see the MTA's support for mailing lists; mailing lists may restrict who can post to the list, restrict the content of what may be posted to the list, add header lines to postings, "tag" the Subject: header line, add disclaimer text, *etc.* But the fundamental and critical difference between a true mailing list and a mail group is the existence of a list owner address -- an address that replaces whatever envelope From was originally present on messages sent to the list address with that list owner address for the postings to the list membership. This has a number of semantic implications, especially as relates to potential

notification messages regarding delivery problems to list members. See Proper use of lists rather than groups for further discussion.
See Also Mail group.

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, *etc.* 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.* See Also Envelope.
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	A 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.

Q

Queue cache database

The queue cache database, also referred to as the Job Controller queue cache, is an in-memory data structure maintained by the Job Controller (as one of its primary responsibilities) that lists the message files awaiting delivery in the MTA's channel queue areas. (In older versions of PMDF, the queue cache database was stored as an on-disk database.)

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 transformating 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 (Simple 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 Messaging Server 4.1 schema (schema tag `nms41`), the Sun Internet Mail Server 4.0 schema (schema tag `sims40`), and the iPlanet Messaging

	Server 5.0 schema (schema tag `ims50`); the MTA merely needs to be told which schema is in use. See in particular the `ldap_schematag` MTA option (and in legacy configuration, also see the `configutil` parameter `local.imta.schematag`). See Also Schema.
Security rule set	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 `PORT_ACCESS` mapping table is used to determine the security rule set to apply to an incoming connection, based on IP addresses and ports.
Sieve	In the context of e-mail, the term "Sieve" usually refers to the Sieve mail filtering language defined in RFC 5228, 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). The Message Store supports use of Sieve tests for specifying which messages to purge.
Sign	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. See Also Certificate, Certificate Authority, TLS (Transport Layer Security).
Slave channel program	Any program which dequeues messages from the MTA's message queues. See Also Master channel program.
SMTP (Simple Mail Transfer Protocol)	See RFC 821, or its update, RFC 5321.
Spam	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.
SSL (Security Sockets Layer)	This protocol was developed by Netscape and has been superseded by TLS, which is backward compatible with SSL.

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 `msgVanityDomain` 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. See Also Domain alias.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

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.