Oracle® Communications Messaging Server

Administration Reference

Release 8.0

July 2015

Oracle Communications Messaging Server Administration Reference, Release 8.0

Contents

Chapter 1. Job Controller Configuration File

Job Controller Configuration File

Topics:

Overview of the Job Controller File

In accordance with the format of the MTA option files, the Job Controller configuration file contains lines of the form:

```
option=value
```

In addition to option settings, the file may contain a line consisting of a section and value enclosed in square-brackets ([ ]) in the form:

```
[section-type=value]
```

Such a line indicates that option settings following this line apply only to the section named by *value*. Initial option settings that appear before any such section tags apply globally to all sections. Per section option settings override global defaults for that section. Recognized section types for the Job Controller configuration file are `POOL`, to define pools and their parameters, and `CHANNEL`, to define channel processing information and `PERIODIC_JOB`. The `PERIODIC_JOB` feature is deprecated and will be removed in a future release. Use `imsched` and the corresponding `local.schedule.* configutil` parameters instead.

Any options permitted on `POOL` or `CHANNEL` sections can be specified at the beginning (general options), thus becoming the default for the option.

The Job Controller configuration file options are described in the following tables (Table 4-24, Table 4-25, Table 4-26). They are split into general options, periodic job options (deprecated), pool options, channel options, and stress options. The stress options are general options (not applicable to pool or channel blocks).

Job Controller General Options

Option	Description

`DEBUG=`*integer*	If `DEBUG` is set to a value other than zero, the MTA writes debugging information to a file in the *msg-svr-base*`/imta/log` directory named `job_controller-`*uniqueid*, where *uniqueid* is a unique ID string that distinctively identifies the file name. The `imsimta purge` utility recognizes the *uniqueids* and can be used to remove older log files. The value for `DEBUG` is a bit mask specifying what sort of debugging information is requested: • `1`---Trace protocol messages between the Job Controller and other MTA components. • `2`---More detailed analysis of the messages and interactions. • `4`---State change events. • `8`---Trace rebuild decisions. • `16`---Dump each queue on every queue action. • `32`---Be cautious about deleting items from queues. • `64`---Perform queue integrity check on every queue operation • `128`---Verbose output about operation of select. Specifying bit value `16` can cause log files to grow very quickly. Specifying `32` does not generate any more output, and should only be used in extreme cases. If `DEBUG` is not specified, it defaults to `0`.
`INTERFACE_ADDRESS=` *IP-address*	Specifies the IP address interface to which the Job Controller should bind. The value specified (adapter) can be one of `ANY`, `ALL`, `LOCALHOST`, or an IP address. By default the Job Controller binds to all addresses (equivalent to specifying `ALL` or `ANY`). Specifying `INTERFACE_ADDRESS=LOCALHOST` means that the Job Controller only accepts connections from within the local machine. This does not affect normal operation, since no inter-machine operation is supported by the Job Controller. However, this may be inappropriate in an HA environment where an HA agent may be checking if the Job Controller is responding. If the machine on which the Messaging Server is running is in an HA environment, has an "internal network" adapter and an "external network" adapter, and you are not confident of your firewall's ability to block connections to high port numbers, you should consider specifying the IP address of the "internal network" adapter.

MAX_CACHE_MESSAGES=*integer*	The Job Controller keeps information about messages in an in-memory structure. The Job controller in-memory cache contains envelope destination and queuing/retry schedule information, not the entire message. The in-memory cache size is approximately 512 bytes per message. In the event that a large backlog builds, it may need to limit the size of this structure. If the number of messages in the backlog exceeds the parameter specified here, information about subsequent messages is not kept in memory. Mail messages are not lost because they are always written to disk, but they are not considered for delivery until the number of messages known by the Job Controller drops to half this number. At this point, the Job Controller scans the queue directory mimicking an `imsimta cache -sync` command. The minimum value is 10. The default is `100000`. This option is used starting in **Messaging Server 7 Update 1** and later.
MAXMESSAGES=*integer*	The Job Controller keeps information about messages in an in-memory structure. In the event that a large backlog builds, it may need to limit the size of this structure. If the number of messages in the backlog exceeds the parameter specified here, information about subsequent messages is not kept in memory. Mail messages are not lost because they are always written to disk, but they are not considered for delivery until the number of messages known by the Job Controller drops to half this number. At this point, the Job Controller scans the queue directory mimicking an `imsimta cache -sync` command. The minimum value is 10. The default is `100000`. This option is used in **Messaging Server 7 Update 1** and prior releases. Starting with **Messaging Server 7 Update 2**, use `MAX_CACHE_MESSAGES`.
REBUILD_IN_ORDER	If set to a non-zero value, then on startup the job controller adds previously untried ZZ* messages to the delivery queue in creation order. Previous (and default) behavior is to add the messages in the order in which they are found on disk. There is a cost associated with recreating the queues in order.
REBUILD_PARALLEL_CHANNELS=*integer*	Specifies how many channel queues are read in parallel on startup and when the job controller is syncing with the disk. Increasing the number can make for a fairer restart when there are massive backlogs, but also has efficiency implications. Default 12.
SECRET=*file_spec*	Shared secret used to protect requests sent to the Job Controller.

`SYNCH_TIME=`*time_spec*	The Job Controller occasionally scans the queue files on disk to check for any new message files that are missing from the Job Controller's list of messages that need to be added. By default, this takes place every four hours, starting four hours after the Job Controller is started. The format of the *time_spec* is *HH*`:`*MM*`/`*hh* `:` *mm* or `/`*hh*`:`*mm*. The variable *hh* `.`*mm* is the interval between the events in hours (*h*) and minutes (*m*). The variable *HH*`:`*MM* is the first time in a day the even should take place. For example specifying, 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.
`TCP_PORT=`*integer*	Specifies the TCP port on which the Job Controller should listen for request packets. Do not change this unless the default conflicts with another TCP application on your system. If you do change this option, change the corresponding `IMTA_JBC_SERVICE` option in the MTA tailor file, *msg-svr-base* `/config/imta_tailor`, so that it matches.

Job Controller `PERIODIC_JOB` Options

The `PERIODIC_JOB` feature is **deprecated** and will be removed in a future release. Use imsched and the corresponding `local.schedule.* configutil` parameters instead.

Option	Description
`COMMAND=` *file_spec*	Specifies the command to be run periodically in a `PERIODIC_JOB` section.
`TIME=` *time_spec*	Specifies the time and frequency that a periodic job is run in a `PERIODIC_JOB` section. By default, this is /4:00, which means every four hours. The format of *time_spec* is *HH*`:`*MM*`/` *hh*:*mm* or `/`*hh*`:`_ mm_. *hh*.*mm* is the interval between the events in hours (*h*) and minutes (*m*). *HH*`:`*MM* is the first time in a day that a job should occur. For example, specifying 15:45/7:15 starts the event at 15:45 and every seven hours and fifteen minutes from then.

Job Controller `POOL` Option

Option	Description
`JOB_LIMIT=` *integer*	Specifies the maximum number of processes that the pool can use simultaneously (in parallel). 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.

Job Controller `CHANNEL` Options

Option	Description
`MASTER_COMMAND=` *file_spec*	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller to run the channel and dequeue messages outbound on that channel. If set outside of a section, it is used as the default by any `CHANNEL` section that doesn't specify a `MASTER_COMMAND`. This option is ignored inside of a `POOL` section.
`MAX_LIFE_AGE=`*integer*	Specifies the maximum life time for a channel master job in seconds. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, `14400` (240 minutes) is used.
`MAX_LIFE_CONNS=` *integer*	In addition to the maximum life age parameter, the life expectancy of a channel master job is limited by the number of times it can ask the Job Controller if there are any messages. If this parameter is not specified for a channel, then the global default value is used. If no default value is specified, `300` is used.
`NONURGENT_DELIVERY=` *time*	Allows the delivery of non-urgent messages to be scheduled only between certain times as configured. The value is a set of up to five time windows separated by commas. Each time window is either a daily window or one of two forms of weekly windows. See below for further details and examples.
`NORMAL_DELIVERY=` *time*	Allows the delivery of non-urgent messages to be scheduled only between certain times as configured. The value is a set of up to five time windows separated by commas. Each time window is either a daily window or one of two forms of weekly windows. See below for further details and examples.
`SLAVE_COMMAND=` *file_spec*	Specifies the full path to the command to be executed by the UNIX system process created by the Job Controller in order to run the channel and poll for any messages inbound on the channel. Most MTA channels do not have a `SLAVE_COMMAND`. If that is the case, the reserved value NULL should be specified. If set outside of a section, it is used as the default by any `CHANNEL` section that doesn't specify a `SLAVE_COMMAND`.
`URGENT_DELIVERY=` *time*	Allows the delivery of non-urgent messages to be scheduled only between certain times as configured. Each time window is either a daily window or one of two forms of weekly windows. See below for further details and examples.

Delivery time schedule parameters.

The `NONURGENT_DELIVERY`, `NORMAL_DELIVERY`, and `URGENT_DELIVERY` parameters allow the delivery of the specified priority messages to be scheduled only between certain times as configured. The value is a set of up to five time windows separated by commas. Each time window is either a daily window, of the form:

```
hh:mm - hh:mm
```

or a weekly window, of one of the following forms:

```
day hh:mm - day hh:mm
day hh:mm - hh:mm
```

In the latter case, the window is assumed to end on the same day that it began. Examples of windows are:

`18:00 - 22:00` which means between 6pm and 10pm each day
`20:00 - 06:30` which means between 8pm and 6:30am each night
`Sat 06:15 - 15:30` which means each Saturday between 6:15am and 3:30pm

`Wed 12:00 - Fri 00:00` which means between noon Wednesday and midnight between Thursday and Friday.

Thus to specify that delivery can be attempted at night or any time on weekends, you could specify:

```
22:00 - 05:30, Sat 00:00 - Sun 23:59
```

Job Controller Stress Options

The "stress" feature was introduced in **Messaging Server 7.0.**

Option	Default - see discussion below
`STRESSBLACKOUT=`*integer*	default 60 seconds
`STRESSTIME=`*integer*	default 120 seconds
`STRESSFACTOR=`*integer*	default is 5
`UNSTRESSFACTOR=`*integer*	defaults to the value of `STRESSFACTOR`
`STRESSJOBS=`*integer*	default is 2
`UNSTRESSJOBS=`*integer*	defaults to the value of `STRESSJOBS`

Starting with Messaging Server 7.0, the Job Controller contains a "stress" feature. The Job Controller can be informed that channels are "stressed" and respond by temporarily reducing delivery jobs on those channels to give the destination some respite. The "stress" feature is important for Message Store delivery channels. Besides accepting new messages, the Message Store is also responsible for responding to end user email client message access requests. Thus, when the Message Store is extraordinarily busy, temporarily reducing the rate of new message deliveries benefits the Message Store. In this situation, the Message Store can focus its resources on staying responsive to end users. By slowing down insertion of new messages into the Message Store, the Message Store is able to respond more quickly to email client access to existing mailboxes and messages.

Message Store delivery channels (`ims-ms` or `tcp_lmtpcs*` channels) automatically inform the Job Controller of stress, when the Message Store detects that it is stressed. You can also use the `imsimta qm stress` command to manually direct the Job Controller to consider any arbitrary channel to be "stressed."

When the job controller receives a channel stress notification, it performs the following steps:

1. Checks to see whether `STRESSBLACKOUT` seconds have elapsed since it last processed a stress notification for the channel. If they have not, it ignores the stress notification.
2. Multiplies the effective thread depth for the channel by `STRESSFACTOR`.
3. Subtracts `STRESSJOBS` from the effective job limit for the channel, but never sets it to less than 1.
4. Stops all delivery jobs for the channel and restarts them according the the new thread depth and job limit.
   Note that this means that a channel can become multiply stressed.

When the job controller does any work for the channel, at anytime, it checks to see if the channel is stressed. If the effective job limit is less than the configured job limit or if the effective thread depth is greater than the configured thread depth, then the channel is considered stressed. If the channel is stressed, and if `STRESSTIME` has elapsed since the last stress notification was processed (or if the imsimta qm unstress command is used), then the job controller does the following:

1. Divides the effective thread depth by `UNSTRESSFACTOR`, but never sets it to less than the configured value.

2. Adds `UNSTRESSJOBS` to the effective job limit, but never sets it to more than the configured value.
3. Starts more processes if the new effective values indicate that they should be started.

As an example, given the following configuration:

```
STRESSFACTOR=10
UNSTRESSFACTOR=3
STRESSJOBS=2
UNSTRESSJOBS=1
THREADDEPTH=5
JOBLIMIT=5
```

the new effective parameters after the first stress indicator would be

```
threaddepth=50
joblimit=3
```

If another stress notification is processed, then the new effective parameters would be

```
threaddepth=500
joblimit=1
```

If no further stress notifications were received, then every `STRESSTIME` seconds, the effective parameters would be changes as follows

```
threaddepth=166
joblimit=2
```

then

```
threaddepth=55
joblimit=3
```

then

```
threaddepth=18
joblimit=4
```

then

```
threaddepth=6
joblimit=4
```

then

```
threaddepth=5
joblimit=5
```

At this point, the job controller no longer considers the channel stressed.

Note that `threaddepth` is configured by the `threaddepth` option on the channel definition in the `imta.cnf` file. And `joblimit` is the smaller of either the `max_jobs` option on the channel in the `imta.cnf` file or the `JOB_LIMIT` on the associated pool in the `job_controller.cnf` file.

When the channel becomes stressed, that does not directly abort delivery attempts. The cause of the stress might also cause deliveries to fail or time out. But that is a result of the root cause of the stress, not an effect of the stress control mechanism. If a backlog is building in the channel queue when it becomes stressed, that backlog continues to grow. As the channel stress reduces and more delivery processes (jobs) are started to service that channel, messages that have not been tried yet begin to dequeue immediately, up to the number of simultaneous operations allowed by the current effective job limit for the channel.

Do not restart the `job_controller` process on the MTA because a restart removes this historical knowledge and therefore causes the maximum number of simultaneous jobs, and therefore delivery attempts, to be started immediately.

Chapter 2. Message Transfer Agent Command-line Utilities

Message Transfer Agent Command-line Utilities

The MTA command-line utilities are used to perform various maintenance, testing, and management tasks for the Message Transfer Agent (MTA).

The MTA commands are also referred to as the `imsimta` commands. The `imsimta` script is located in the *msg-svr-base*`/` directory.

msg-svr-base represents the directory path in which you install the server.

Topics:

- MTA Commands
- Command Descriptions

MTA Commands

The commands are listed in the following table.

MTA Commands

Command	Description
imsimta cache	Performs operations on the queue cache.
imsimta chbuild	Compiles the MTA character set conversion tables.
imsimta cnbuild	Compiles the MTA configuration files.
imsimta counters	Performs operations on the channel counters.
imsimta crdb	Creates an MTA database.
imsimta find	Locates the precise filename of the specified version of an MTA log file
imsimta kill	Terminates the specified process.
imsimta process	Lists currently running MTA jobs.
imsimta program	Manipulates the MTA program delivery options.
imsimta purge	Purges MTA log files.
imsimta qclean	Holds or deletes message files containing specific substrings in their envelope From:address, Subject: line, or content.

imsimta qm	Manages MTA message queues.
imsimta qtop	Displays the most frequently occurring envelope From: Subject:, or message content fields found in message files in the channel queues.
imsimta refresh	Combines the functionality of the imsimta cnbuild and imsimta restart utilities.
imsimta reload	Allows changes to certain configuration files to take effect without restarting the server.
imsimta renamedb	Renames a MTA database.
imsimta restart	Restarts detached MTA processes.
imsimta return	Returns (bounces) a mail message to its originator.
imsimta run	Processes messages in a specified channel.
imsimta shutdown	Shuts down the MTA Job Controller and the MTA Dispatcher as well as individual processes running under the Dispatcher.
imsimta start	Starts the MTA Job Controller and Dispatcher.
imsimta stop	Shuts down the MTA Job Controller and the MTA Dispatcher as well as individual processes running under the Dispatcher.
imsimta submit	Processes messages in a specified channel.
imsimta test	Performs tests on mapping tables, wildcard patterns, address rewriting, and URLs.
imsimta test -domain	Verifies the validity of domain structures in the directory and displays domain-related information.
imsimta version	Prints the MTA version number.
imsimta view	Displays log files.

Command Descriptions

You need to be logged in as `root` (UNIX) or administrator (Windows NT) to run the MTA commands. Unless mentioned otherwise, all MTA commands should be run as `mailsrv` (the mail server user that is created at installation).

imsimta cache

The MTA maintains an in-memory cache of all the messages currently stored in its queues. This cache is called the queue cache. The purpose of the queue cache is to make dequeue operations perform more efficiently by relieving master programs from having to open every message file to find out which message to dequeue and in which order.

Syntax

```
imsimta cache -change -parameter| -dump | -sync[hronize] | -view <channel> |
-walk
```

Options

The options for this command are:

Option	Description
`-change -global -debug=`*integer*	Allows debugging for the job controller on the fly.
`-change -global -max_messages=` *integer*	Allows setting the maximum number of messages for the job controller on the fly.
`-change -channel_template=` *name* `master_job=` *command*	Allows changing the channel template for the job controller on the fly. Changing parameters for a channel template (e.g., tcp_*) changes that parameter for all channels derived from that template.
`-change -channel_template=` *name* `slave_job=` *command*	Allows changing the channel template for the job controller on the fly. Changing parameters for a channel template (e.g., tcp_*) changes that parameter for all channels derived from that template.
`-change -channel=` *name* `master_job=` *command*	Allows changing the master job for the job controller on the fly.
`-change -channel=` *name* `slave_job=` *command*	Allows changing the slave job for the job controller on the fly.
`-change -channel=` *name* `thread_depth=` *integer*	Allows setting the thread depth for the job controller on the fly.
`-change -channel=` *name* `job_limit=` *integer*	Allows setting the job limit for the job controller on the fly.
`-dump`	Provides the same output as the `-view` switch with no channel specified.
`-sync`	Updates the active queue cache by updating it to reflect all non-held message files currently present in the *msg-svr-base*`/imta/queue/` subdirectories. Note that the `-sync` option does not remove entries from the queue cache. The queue cache entries not corresponding to an actual queued message are silently discarded by channel master programs.
`-view` <*channel*>	Shows the current non-held entries in the MTA queue cache for a channel. <*channel*> is the name of the channel for which to show entries e.g. *tcp_local* This is a potentially expensive command if you have a large backlog of messages. If a <*channel*> is not specified, entries in all channels will be displayed.
`-walk [-debug=xxx]`	Used to diagnose potential problems with the job controller. Each time this command is run, the internal state of the message queues and job scheduling information is written to the `job_controller.log` file. In addition, the job controller debug mask is set to the value given. You should not run this command unless you are instructed to do so by support.

Examples

To synchronize the queue cache:

```
imsimta cache -sync
```

To view entries in the queue cache for the `tcp_local` channel, execute the command:

```
imsimta cache -view tcp_local
```

imsimta chbuild

The `imsimta chbuild` command compiles the character set conversion tables and loads the resulting image file into shared memory. The MTA ships with complete character set tables so you would not normally need to run this command. You would use `imsimta chbuild` only if you added or modified any character sets.

Syntax

```
imsimta chbuild [-image_file=<file_spec> | -noimage_file]
[-maximum | -nomaximum] [-option_file=[<option_file>]
| -nooption_file] [-remove] [-sizes |-nosizes] [-statistics |
-nostatistics]
```

Options

The options for this command are:

Option	Description
`-image_file=` *file_spec* | `-noimage_file`	By default, `imsimta chbuild` creates as output the image file named by the `IMTA_CHARSET_DATA` option of the MTA tailor file, *msg-svr-base* `/config/imta_tailor`. With the `-image_file` option, an alternate file name may be specified. When the `-noimage_file` option is specified, `imsimta chbuild` does not produce an output image file. The `-noimage_file` option is used in conjunction with the `-option_file` option to produce as output an option file that specifies table sizes adequate to hold the tables required by the processed input files.
`-maximum` | `-nomaximum`	The file *msg-svr-base*`/config/maximum_charset.dat` is read in addition to the file named by the `IMTA_CHARSET_OPTION_FILE` option of the MTA tailor file, *msg-svr-base*`/config/imta_tailor`, when `-maximum` is specified. This file specifies near `-maximum` table sizes but does not change any other settings. Use this option only if the current table sizes are inadequate. The `-noimage` and `-option_file` options should always be used in conjunction with this option – 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 build a properly sized option file for use in building a manageable configuration with a subsequent `imsimta chbuild` invocation.
`-option_file=[` *option_file*`]` | `-nooption_file`	`imsimta chbuild` can produce an option file that contains the correct table sizes to hold the conversion tables that were just processed (plus a little room for growth). The `-option_file` option causes this file to be output. By default, this file is the file named by the `IMTA_CHARSET_OPTION_FILE` option of the MTA tailor file, *msg-svr-base*`/config/imta_tailor`. The value of the `-option_file` option may be used to specify an alternate file name. If the `-nooption_file` option is given, then no option file is output. `imsimta chbuild` always reads any option file (for example, the file named by the `IMTA_OPTION_FILE` option of the MTA tailor file) that is already present; use of this option does not alter this behavior. However, use of the `-maximum` option causes `imsimta chbuild` to read options from `maximum_charset.dat` in addition to `IMTA_CHARSET_OPTION_FILE`. This file specifies near-maximum table sizes. Use this option only if the current table sizes are inadequate, and only use it to create a new option file. The `-noimage_file` option should always be specified with `-maximum`, since a maximum-size image would be enormous and inefficient.
`-remove`	Removes any existing compiled character set conversion table. This is the file named by the `IMTA_CHARSET_DATA` option of the MTA tailor file, *msg-svr-base* `/config/imta_tailor`.
`-sizes` | `-nosizes`	The `-sizes` option instructs `imsimta chbuild` to output or suppress information on the sizes of the uncompiled conversion tables. The `-nosizes` option is the default.
`-statistics` | `-nostatistics`	The `-statistics` option instructs `imsimta chbuild` to output or suppress information on the compiled conversion tables. This information gives a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the `-option_file` option is needed. The `-nostatistics` option is the default.

Example

The standard command you use to compile character set conversion tables is:

```
imsimta chbuild
```

imsimta cnbuild

The `imsimta cnbuild` command compiles the textual configuration, option, mapping, conversion, circuit check and alias files, and loads the resulting image file into shared memory. The resulting image is saved to a file usually named `imta/lib/config_data` by the `IMTA_CONFIG_DATA` option of the MTA tailor file, *msg-svr-base*/`config/imta_tailor`.

Whenever a component of the MTA (for example, a channel program) must read a compiled configuration component, it first checks to see whether the file named by the MTA tailor file option `IMTA_CONFIG_DATA` is loaded into shared memory; if this compiled image exists but is not loaded, the MTA loads it into shared memory. If the MTA finds (or not finding, is able to load) a compiled image in shared memory, the running program uses that image.

The reason for compiling configuration information is simple: performance. The only penalty paid for compilation is the need to recompile and reload the image any time the underlying configuration files are edited. Also, be sure to restart any programs or channels that load the configuration data only once when they start up-for example, the MTA multithreaded SMTP server.

It is necessary to recompile the configuration every time changes are made to any of the following files:

- MTA configuration file (or any files referenced by it)
- MTA system alias file
- MTA mapping file
- MTA option file
- MTA conversion file
- MTA security configuration file
- MTA circuit check configuration file
- MTA system wide filter file

Specifically, these are the files pointed at by the MTA tailor file options `IMTA_CONFIG_FILE`, `IMTA_ALIAS_FILE`, `IMTA_MAPPING_FILE`, `IMTA_OPTION_FILE`, and `IMTA_CONVERSION_FILE` respectively, which usually point to the following files:

- *msg-svr-base*`/config/imta.cnf`
- *msg-svr-base*`/config/aliases`
- *msg-svr-base*`/config/mappings`
- *msg-svr-base*`/config/option.dat`
- *msg-svr-base*`/config/conversions`

Note
Until the configuration is rebuilt, changes to any of these files are not visible to the running MTA system.

Syntax

```
imsimta cnbuild [-image_file=<file_spec> | -noimage_file]
[-maximum | -nomaximum] [-option_file=[<option_file>]
| -nooption_file] [-remove] [-sizes | -nosizes] [-statistics |
-nostatistics]
```

Options

The options for this command are:

Option	Description
`-image_file=` *file_spec* | `-noimage_file`	By default, `imsimta cnbuild` creates as output the image file named by the `IMTA_CONFIG_DATA` option of the MTA tailor file, *msg-svr-base* `/config/imta_tailor`. With the `-image_file` option, an alternate filename can be specified. When the `-noimage_file` option is specified, `imsimta cnbuild` does not produce an output image file. This option is used in conjunction with the `-option_file` option to produce as output an option file which specifies table sizes adequate to hold the configuration required by the processed input files. The default value is `-image_file=IMTA_CONFIG_DATA`.
`-maximum` | `-nomaximum`	*msg-svr-base*`/config/maximum.dat` is read in addition to the file named by the `IMTA_OPTION_FILE` option in the MTA tailor file, *msg-svr-base* `/config/imta_tailor`. This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this option if the current table sizes are inadequate. The `-noimage` and `-option_file` options should always be used in conjunction with this qualifier; 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 build a properly-sized option file so that a proportionately-sized configuration can be built with a subsequent `imsimta cnbuild` invocation. The default is `-nomaximum`.
`-option_file=[` *option_file*`]` | `-nooption_file`	`imsimta cnbuild` can optionally 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` option causes this file to be output. By default, this file is the file named by the `IMTA_OPTION_FILE` option in the MTA tailor file, *msg-svr-base*`/config/imta_tailor`. The value on the `-option_file` option may be used to specify an alternate file name. If the `-nooption_file` option is given, then no option file will be output. `imsimta cnbuild` always reads any option file that is already present via the `IMTA_OPTION_FILE` option of the MTA tailor file, *msg-svr-base*`/config/imta_tailor`; use of this option will not alter this behavior. However, use of the `-maximum` option causes `imsimta cnbuild` to read MTA options from the *msg-svr-base*`/config/maximum.dat` file in addition to reading the file named by `IMTA_OPTION_FILE`. This file specifies near maximum table sizes. Use this option only if the current table sizes are inadequate, and only to create a new option file. The `-noimage_file` option should always be specified when `-maximum` is specified since a maximum-size image would be enormous and wasteful. The default value is `-option_file=IMTA_OPTION_FILE`.
`-remove`	Remove any existing compiled configuration; for example, remove the file named by the `IMTA_CONFIG_DATA` option of the MTA tailor file, *msg-svr-base* `/config/imta_tailor`.
`-sizes` | `-nosizes`	The `-sizes` option instructs `imsimta cnbuild` to output information on the sizes of uncompiled MTA tables. The `-nosizes` option is the default.
`-statistics` | `-nostatistics`	The `-statistics` option instructs `imsimta cnbuild` to output information table usage. This information gives a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the `-resize_tables` option is needed. The `-nostatistics` option is the default.

Examples

To regenerate a compiled configuration enter the following command:

```
imsimta cnbuild
```

After compiling the configuration, restart any programs that may need to reload the new configuration. For example, the SMTP server should be restarted:

```
imsimta restart dispatcher
```

Note
`imsimta cnbuild` is executed whenever the `imsimta refresh` command is invoked.

imsimta counters

The MTA accumulates message traffic counters for each of its active channels. These statistics, referred to as channel counters, are kept in shared memory. The `imsimta counters` command manipulates these counters.

Syntax

```
imsimta counters -clear

imsimta counters -create [-max_channels=<value>]

imsimta counters -delete

imsimta counters -show [-associations | -noassociations] [-channels |
-nochannels] [-headers | -noheaders] [-output=<file_spec>]
```

Options

The options for this command are:

Option	Description
`-associations` | `-noassociations`	Specifies whether or not to show the in-memory cache of association counters. The `-associations` option is the default. This option is only used with the `-show` option.
`-channels` | `-nochannels`	Specifies whether or not to show the in-memory cache or channel counters. The `-channels` option is the default. This option is only used with the `-show` option.
`-clear`	The `-clear` command clears the in-memory channel counters.
`-create`	Creates the in-memory channel counters. Counters are not created by default. You must create the counters if you wish to use them. You must create them after restarting the MTA as well.
`-headers` | `-noheaders`	Controls whether or not a header line describing each column in the table of counters is output. The `-headers` option is the default. This option is only used with the `-show` option.
`-max_channels=`*value*	By default, the in-memory channel counters can hold information for `CHANNEL_TABLE_SIZE` channels. `CHANNEL_TABLE_SIZE` is the value specified by the MTA option file option of the same name. Use the `-max_channels=`*value* option to select a different size. This option is used only with the `-create` option.
`-delete`	Deletes the in-memory channel counters.
`-show`	Displays the in-memory channel counters.
`-headers` | `-noheaders`	Controls whether or not a header line describing each column in the table of counters is output. The `-headers` option is the default. This option is only used with the `-show` option.
`-output=`*file_spec*	Directs the output to the specified file. By default, the output appears on your display. This option is only used with the `-show` option.

Examples

To display the counters for all channels:

```
imsimta counters -show
```

imsimta crdb

The `imsimta crdb` command creates and updates MTA database files. `imsimta crdb` converts a plain text file into MTA database records; from them, it either creates a new database or adds the records to an existing database.

In general, each line of the input file must consist of a left side and a right side. The two sides are separated by one or more spaces or tabs. The left side is limited to 32 characters in a short database (the default variety) and 80 characters in a long database. The right 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 side unless the `-quoted` option is specified. Comment lines may be included in input files. A comment line is a line that begins with an exclamation mark (!) in column 1.

Syntax

```
imsimta crdb <input-file-spec> <output-database-spec> [-append | -noappend]
[-count | -nocount] [-duplicates | -noduplicates] [-long_records |
-nolong_records] [-quoted | -noquoted] [-remove | -noremove] [-statistics |
-nostatistics] [-strip_colons | -nostrip_colons]
```

Options

The options for this command are:

Option	Description
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. This attribute is mandatory.
output-database-spec	The initial name string of the files to which to write the database (unless `-dump` is specified). The `.db` extension is appended to the file name. This attribute is mandatory.
`-append` | `-noappend`	When the default, `-noappend`, option is in effect, a new database is created, overwriting any old database of that name. Use the `-append` option to instruct the MTA to instead add the new records to an existing database. The `-noappend` option is the default. In the event of a duplicate record, the newly appended record overwrites the old record when `-noduplicates` is specified.
`-count` | `-nocount`	Controls whether or not a count is output after each group of 100 input lines are processed. The `-count` option is the default.
`-duplicates` | `-noduplicates`	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. The `-noduplicates` option is the default.
`-long_records` | `-nolong_records`	Controls the size of the output records. By default, left sides are limited to 32 characters and right sides are limited to 80 characters. If `-long_records` is specified, the limits are changed to 80 and 256, respectively. The `-nolong_records` option is the default.
`-quoted` | `-noquoted`	Controls the handling of quotes. Normally `imsimta crdb` pays no 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 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. The quotes are not removed unless the `-remove` option is also specified. The `-noquoted` option is the default.
`-remove` | `-noremove`	Controls 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 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. `-remove` is ignored if `-quoted` is not in effect. The `-noremove` option is the default.
`-statistics` | `-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. The `-statistics` option is the default.
`-strip_colons` | `-nostrip_colons`	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. The `-nostrip_colons` is the default.

Example

The following commands create an alias database with "long" record entries. 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.

```
imsimta crdb -long_records aliases-tmp

imsimta renamedb aliases-tmp IMTA_ALIAS_DATABASE
```

imsimta crdb -dump

The `imsimta crdb -dump` command writes the entries in MTA databases to a flat ASCII file. In particular, this command may be used to write the contents of an old style database to a file from which a new style database may be built using the `imsimta crdb` command. The output begins with a comment line that displays a proper `imsimta crdb` command to use in order to return the ASCII output to a database.

Note
Make sure you are logged in as `mailsrv` (the mail server user) before performing this command.

Syntax

```
imsimta crdb -dump <input-database-spec> [<output-file-spec>]
```

Parameters

The parameters for this command are:

Parameter	Description
input-database-spec	Database from which to read entries. By default, the MTA looks for a current format database of the given name; if this does not exist, the MTA will look for an old format database of the given name. The special keywords `IMTA_ALIAS_DATABASE`, `IMTA_REVERSE_DATABASE`, and `IMTA_GENERAL_DATABASE` are supported; the use of such a special keyword tells the MTA to dump the database specified by the corresponding MTA tailor file option.
output-file-spec	ASCII file to which the entries stored in the database are written. This file should be in a directory where you have write permissions. If an output file is not specified, the output is written to `stdout`.

Examples

The following command can be used to dump the contents of an alias database to a file, and then to recreate the alias database from that file.

```
imsimta crdb -dump IMTA_ALIAS_DATABASE alias.txt
imsimta crdb alias.txt alias-tmp
imsimta renamedb alias-tmp IMTA_ALIAS_DATABASE
```

imsimta find

The `imsimta find` utility locates the precise filename of the specified version of an MTA log file. MTA log files have a `-`*uniqueid* appended to the filename to allow for the creation of multiple versions of the log file. On UNIX, the `-`*uniqueid* is appended to the very end of the filename (the end of the file extension), while on Windows NT, the `-`*uniqueid* is appended to the end of the name part of the filename, before the file extension. The `imsimta find` utility understands these unique ids and can find the particular filename corresponding to the requested version of the file.

Syntax

```
imsimta find <file-pattern> [-f=<offset-from-first>] [-l=<offset-from-last>]
```

Options

The options for this command are:

Option	Description
`-f=` *offset-from-first*	Finds the specified version of the file (starting from 0). For example, to find the earliest (oldest) version of the file, specify `-f=0`. By default, `imsimta find` finds the most recent version of the file.
`-l=` *offset-from-last*	Finds the last version of the specified file. For example, to find the most recent (newest) version of the file, specify `-l=0`. By default, `imsimta find` finds the most recent version of the file.
file-pattern	Specifies a filename pattern for which the log file to find.

Examples

The following command prints out the filename of the `tcp_local_slave.log-`*uniqueid* file most recently created:

```
imsimta find <msg-svr-base>/data/log/tcp_local_slave.log
```

The following command displays the filename of the oldest tcp_bitnet_master.log-uniqueid file:

```
imsimta find <msg-svr-base>/data/log/tcp_bitnet_master.log -f=0
```

imsimta kill

The `imsimta kill` utility immediately and indiscriminately terminates the specified process. This command is equivalent to the UNIX `kill -9` command. The process is terminated even if it is in the middle of transferring email. So use of the `imsimta shutdown` utility, which performs an orderly shutdown, is generally preferable.

Syntax

```
imsimta kill <component>
```

Note
You must have the same process id as the process to be killed, or be `root`. This utility is not available on Windows NT.

component is the MTA component to be killed. Valid values are `job_controller` and `dispatcher`.

imsimta process

This command displays the current MTA processes. Additional processes may be present if messages are currently being processed, or if certain additional MTA components are in use.

Syntax

```
imsimta process
```

Example

The following command shows current MTA processes:

```
# imsimta process
imsimta process

USER PID S VSZ RSS STIME TIME COMMAND
mailsrv 15334 S 21368 9048 17:32:44 0:01 imta/bin/dispatcher
mailsrv 15337 S 21088 10968 17:32:45 0:01 imta/bin/tcp_smtp_server
mailsrv 15338 S 21080 11064 17:32:45 0:01 imta/bin/tcp_smtp_server
mailsrv 15349 S 21176 10224 17:33:02 0:02 imta/bin/job_controller
```

imsimta program

The `imsimta program` command is used to manipulate the program delivery options.

This command can be executed as `root` or `mailsrv`. `mailsrv` is the default user for Messaging Server, but could be whatever the specified user name for the Messaging Server is when Messaging Server is installed.

The program is passed the entire message, unparsed from `stdin`. This includes the From line (without the colon) as the first line, followed by the headers and the message body. This may include any MIME attachments that are part of the message.

Syntax

```
imsimta program -a -m <method> -p <program>
[-g <argument_list>] [-e <exec_permission>]
imsimta program -d -m <method>

imsimta program -c -m <method> -p <program>
| -g <argument_list> | -e <exec_permission>
```

Options

The options for this command are:

Option	Description
`-a`	Add a method to the set of program delivery methods. This option cannot be used with the `-d`, `-c`, `-l`, or `-u` options.
`-c`	Change the arguments to a program that has already been entered.
`-m` *method*	Name given by the administrator to a particular method. This will be the name by which the method will be advertised to users. Method names must not contain spaces, tabs, or equal signs (=). The method name cannot be none or local. The method name is restricted to U.S. ASCII. This option is required with the `-a`, `-d`, `-c`, and `-u` options.
`-p` *program*	Actual name of the executable for a particular method. The executable should exist in the programs directory (*msg-svr-base*`/data/site-programs`) for the add to be successful. It can be a symbolic link to an executable in some other directory. This option is required with the `-a` option.
`-g` *argument_list*	Argument list to be used while executing the program. If this option is not specified during an add, no arguments will be used. Each argument must be separated by a space and the entire argument list must be given within double quotes. If the %s tag is used in the argument list, it will be substituted with the user's username for programs executed by the users and with *username+programlabel* for programs executed by inetmail. *programlabel* is a unique string to identify that program. This option can be used with the `-a` and `-c` options.
`-e` *exec_permission*	*exec_permission* can be `user` or `postmaster`. If it is specified as user, the program is executed as the user. By default, execute permission for all programs are set to postmaster. Programs with *exec_permission* set to user can be accessed by users with UNIX accounts only. This option can be used with the `-a` and `-c` options. The directory from where this program is run as postmaster is the postmaster's home directory. If specified as `user`, then the user's home directory is the environment where the program is run as the user.
`-d`	Delete a method from the list of supported program delivery methods. This option cannot be used with the `-a`, `-c`, `-l`, or `-u` options.
`-h`	Help for this command.
`-l`	List all methods.
`-u`	List all users using the method specified with the `-m` option.

Examples

To add a method `procmail1` that executes the program `procmail` with the arguments `-d` *username* and executes as the user, enter the following:

```
imsimta program -a -m procmail1 -p procmail -g "-d %s" -e user
```

imsimta purge

The `imsimta purge` command deletes older versions of MTA log files. `imsimta purge` can determine the age of log files from the uniqueid strings terminating MTA log file names.

Syntax

```
imsimta purge [<file-pattern>] -day=<dvalue> -hour=<hvalue> -num=<nvalue>
```

Options

The options for this command are:

Option	Description
file-pattern	If specified, the *file-pattern* parameter is a file name pattern that establishes which MTA log files to purge. The default pattern, if none is specified, is `log/imta/log`.
`-day=` *dvalue*	Purges all but the last *dvalue* days worth of log files.
`-hour=` *hvalue*	Purges all but the last *hvalue* hours worth of log files.
`-num=` *nvalue*	Purges all but the last *nvalue* log files. The default is 5.

Example

To purge all but the last five versions of each type of log file in the `log/imta` directory:

```
imsimta purge
```

imsimta qclean

The `imsimta qclean` utility holds or deletes message files containing specific substrings in their envelope From:address, Subject: line, or content.

Syntax

```
imsimta qclean
[-content=<substring>] [-from=<substring>][-subject=<substring>]
[-to=<substring>] [-domain_to=<substring>] [-database] [-delete | -hold]
[-directory_tree] [-ignore_zz] [-match=<keyword>] [-min_length=<n>]
[-threads | -nothreads] [-verbose | -noverbose] [<channel>]
```

Options

The options for this command are:

Option	Description
`-content=` *substring* `-from=` *substring* `-subject=` *substring* `-to=` *substring* `-domain_to=` *substring*	Specifies the substrings for which to search. Any combination of `-content`, `-from`, `-subject`, `-to`, and `-domain_to` may be specified. However, only one of each may be used. When a combination of such options is used, the `-match` option controls whether the options are interpreted as further restrictions (`-match=AND`) or as alternatives (`-match=OR`).The `-domain_to` option scans for frequently occurring envelope To: addresses. Identical to the `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.
`-database`	Specifies that only message files identified by the queue cache is searched.
`-delete`	Deletes matching message files.
`-hold`	Holds matching message files.
`-directory_tree`	Searches all message files that are actually present in the channel queue directory tree.
`-ignore_zz`	Ignores queued message files with file names beginning with "ZZ". This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
`-match=`*keyword*	Controls whether a message file must contain all (`-match=AND`) or only one of (`-match=OR`) the specified substrings in order to be held or deleted. The default is `-match=AND`.
`-min_length=`*n*	Specifies the minimum length of the substring for which to search. By default, each substring must be at least 24 bytes long. Use the `-min_length` option to override this limit.
`-threads=`*n* | `-nothreads`	Accelerates the searching on multiprocessor systems by dividing the work amongst multiple, simultaneous running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be an integer between 1 and 8. The default is `-nothreads`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies an MTA channel area to be searched for matching messages. The * or ? wildcard characters may be used in the channel specification.

imsimta qm

The `imsimta qm` utility inspects and manipulates the channel queues and the messages contained in the queues. `imsimta qm` contains some functionality overlap with the `imsimta cache` and `imsimta counters` commands.

For example, some of the information returned by `imsimta cache -view` is also available through the `imsimta qm directory` command. However, `imsimta qm`, does not completely replace `imsimta cache` or `imsimta queue`.

You must be `root` or `mailsrv` to run `imsimta qm`.

`imsimta qm` can be run in an interactive or non-interactive mode. To run `imsimta qm` in the interactive mode, enter:

```
imsimta qm
```

You can then enter the sub-commands that are available for use in the interactive mode. To exit out of the interactive mode, enter `exit` or `quit`.

To run `imsimta qm` in the non-interactive mode, enter:

```
imsimta qm <sub-commands> [<options>]
```

Note that some of the sub-commands available in the interactive mode are not available in the non-interactive mode, and vice versa. Refer to the following sub-commands for availability in each mode.

Sub-Commands

clean

The `clean` sub-command holds or deletes message files containing specific substrings in their envelope From: address, Subject: line, or content.

Available in both interactive and non-interactive modes.

```
clean [-content=<substring>] [-from=<substring>] [-subject=<substring>]
[-to=<substring>] [-domain_to=substring]
[-database | -directory_tree] [-delete | -hold] [-ignore_zz]
[-match=<keyword>] [-min_length=<n>] [-threads=<n> | -nothreads]
[-verbose | -noverbose] [<channel>]
```

The options for this sub-command are:

Option	Description
`-content=` *substring* `-from=` *substring* `-subject=` *substring* `-to=` *substring* `-domain_to=` *substring*	Specifies the substrings for which to search. Any combination of each option may be used. However, only one of each may only be used. When a combination of such options is used, the `-match` option controls whether the options are interpreted as further restrictions (`-match=AND`), or as alternatives (`-match=OR`).The `-domain_to` option scans for frequently occurring envelope To: addresses. Identical to the `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.The `-from` option can take an empty address using, for example, `imsimta qm clean -from=\<\>`.
`-database` | `-directory_tree`	Controls whether the message files searched are only those with entries in the queue cache (`-database`) or all message files actually present in the channel queue directory tree (`-directory_tree`). When neither `-database` nor `-directory_tree` is specified, then the view selected with the `view` sub-command will be used. If no `view` sub-command has been issued, then `-directory_tree` is assumed.
`-delete` | `-hold`	Specifies whether matching message files are held (`-hold`) or deleted (`-delete`). The `-hold` option is the default.
`-ignore_zz`	Ignores queued message files with file names beginning with "ZZ". This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
`-match=`*keyword*	Controls whether a message file must contain all (`-match=AND`) or only one of (`-match=OR`) the specified substrings in order to be held or deleted. The substrings are specified by the `-content`, `-env_from`, and `-subject` options. The default is `-match=AND`.
`-min_length=`*n*	Overrides the length limit for each substring to be searched. By default, the limit is 24 bytes (`-min_length=24`).
`-threads=`*n* | `-nothreads`	Accelerates the searching on multiprocessor systems by dividing the work amongst multiple, simultaneous running threads. To run *n* simultaneous searching threads, specify `-threads=`*n*. The value *n* must be an integer between 1 and 8. The default is `-nothreads`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies a specific MTA channel area to be searched for matching messages. The * or ? wildcard characters may be used in the channel specification.

counters clear

The `counters clear` sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and association counters if the segment does not already exist.
2. Sets all counter values to zero.
3. When `-channels` is specified, sets the counts of stored messages, recipients, and volume from the queue cache database.

Available for both interactive and non-interactive modes.

```
counters clear [-channels] [-associations]
```

The options for this sub-command are:

Option	Description
`-channels`	Clears the message counters
`-associations`	Clears the association counters

When neither option is specified, both are assumed. When `-associations` is specified and `-channels` is not specified, step (3) is not performed.

counters create

The `counters create` sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and association counters if the segment does not already exist.
2. Sets the counts of stored messages, recipients, and volume from the queue cache database.

Available for both interactive and non-interactive modes.

```
counters create [-max_channels=<n>]
```

The option for this sub-command is:

Option	Description
`-max_channels` =*n*	Tells the MTA how many channels to allow for in the memory segment. If this option is omitted, then the MTA looks at the `imta.cnf` file and determines a value on its own.

counters delete

The `counters delete` sub-command deletes the shared memory segment used for channel message and association counters. Note that active MTA server processes and channels will likely recreate the memory segment.

Available for both interactive and non-interactive modes.

```
counters delete
```

counters show

Use the `counters show` sub-command to display channel message counters. When the optional *channel-name* parameter is omitted, * (wildcard) is assumed and the message counters for all channels are displayed. The *channel-name* parameter may contain the * and? wildcard characters.

The *counters show* sub-command performs the following operations:

1. Creates the shared memory segment for the channel message and associated counters if the segment does not already exist.
2. Sets the counts of stored messages, recipients, and volume from the queue cache database.
3. Displays the message counters for the specified channels.

Available for both interactive and non-interactive modes.

```
counters show [-headers] [-noheaders] [-output=<file-spec>] [<channel-name>]
```

The options for this sub-command are:

Option	Description
`-headers` or `-noheaders`	Controls whether or not a heading is displayed. The `-headers` option is the default.
`-output=`*file_spec*	Causes the output to be written to a file. Any existing file with the same name as the output file is overwritten.

date

Displays the current time and date in RFC 822, 1123 format.

Available for both interactive and non-interactive modes.

```
date
```

delete

Deletes the specified messages displayed in the most recently generated message queue listing.

```
delete [-channel=<name> [-all]] [-confirm | -noconfirm]
[-log | -nolog] [<id>...]
```

The *id* parameter specifies the messages to be deleted.

See imsimta qm Options for information on using the `-channel`, `-all`, `-confirm`, and `-log` options.

Available only in interactive mode.

directory

Generates a listing of queued message files. By default, the `imta/queue` directory tree is used as the source of queued message information; this default may be changed with the view sub-command. The `-database` and `-directory_tree` options may be used to override the default.

Available for both interactive and non-interactive modes.

```
directory [-held | -noheld] [-database] [-directory_tree]
[-envelope] [-owner=<username>] [-from=<address>] [-to=<address>]
[-match=<bool>] [-file_info | -nofile_info] [-total | -nototal]
[<channel-name>]
```

The options for this sub-command are:

Option	Description
`-database`	Obtains message information from the Job Controller.
`-directory_tree`	Selects the on-disk directory tree as the source of message information.
`-envelope`	Generates a listing which also contains envelope address information.
`-total` | `-nototal`	Generates size and count totals across all selected channels.
`-owner=`*username*	Lists only those messages owned by a particular user. Messages enqueued by a local user will be owned by that user; most other messages will be owned by `mailsrv`. Use of the `-owner` option implies `-database`.
`-from=`*address* and `-to=`*address* and `-match=`*bool*	Lists only those messages with envelope From: or To: addresses matching the specified address. When both `-from` and `-to` are specified, a message is listed if either its envelope From: or To: addresses match the specified addresses. This corresponds to the `-match=or` option. Specify `-match=and` to list only messages matching both the specified From: and To: addresses. Use of `-from` or `-to` implies `-envelope`.*address* can include a wild card (*) that matches a sequence of characters or a % character that matches a single character.
`-held` | `-noheld`	By default, active messages are listed. Specify `-held` to instead list messages which have been marked as held. Note that `-held` implies `-directory_tree`.
`-file_info` | `-nofile_info`	When the directory tree is scanned, each message file is accessed to determine its size as measured in units of blocks (normally 1024 bytes). To suppress this behavior and speed up generation of the listing, specify `-nofile_info`. When the queue cache database is used, the `-nofile_info` option is ignored as the size information is stored in the database.
channel-name	Restricts the listing to one or more channels. If the *channel-name* parameter is omitted, a listing is made for all channels. The channel name parameter may contain the * and ? wildcard characters.

exit

Exits the `imsimta qm` utility. Synonymous with the `quit` sub-command.

Available for both interactive and non-interactive modes.

```
exit
```

held

Generates a listing of message files which have been marked as held. This listing is always generated from the `imta/queue/` directory tree.

Available for both interactive and non-interactive modes.

```
held [-envelope] [-file_info | -nofile_info] [-total | -nototal]
[-from=<address>] [-to=<address>] [-match=<bool>] [<channel-name>]
```

The options for this sub-command are:

Option	Description
`-envelope`	Generates a listing which also contains envelope address information.
`-total`| `-nototal`	Generate size and count totals across all selected channels.
`-from=`*address* and `-to=`*address* and `-match=`*bool*	Lists only those messages with envelope From: or To: addresses matching the specified address. When both `-from` and `-to` are specified, a message is listed if either its envelope From: or To: addresses match the specified addresses. This corresponds to the `-match=or` option. Specify `-match=and` to list only messages matching both the specified From: and To: addresses. Use of `-from` or `-to` implies `-envelope`.
`-file_info`| `-nofile_info`	When the directory tree is scanned, each message file is opened to determine its size as measured in units of blocks (normally 1024 bytes). To suppress this behavior and speed up generation of the listing, specify `-nofile_info`.
channel-name	Restricts the listing to one or more channels. If the *channel-name* parameter is omitted, a listing is made for all channels. The *channel-name* parameter may contain the * and ? wildcard characters.

history

Displays any delivery history information for the specified messages from the most recently generated message queue listing.

Available only in interactive mode.

```
history [-channel=<name> [-all] ] [-confirm | -noconfirm] [<id>...]
```

Use the *id* parameter to specify the messages whose history is displayed.

See imsimta qm Options for information on using the `-channel`, `-all`, and `-confirm` options.

hold

Marks as held the specified messages from the most recently generated message queue listing

Available only in interactive mode.

```
hold [-channel=<name> [-all]] [-confirm | -noconfirm]
[-log | -nolog] [<id>...]
```

Use the *id* parameter to specify the messages to mark as held.

See imsimta qm Options for information on the `-channel`, `-all`, `-confirm`, and `-log` options.

jobs

The `imsimta qm jobs` utility displays what messages are being processed by what jobs for what channels.

```
jobs
```

Example of output:

```
channel <channel name>
job <pid>
host <host name>
host <host name>
<count of hosts> HOST BEING PROCESSED BY JOB <pid>
message <subdir/message name>
message <subdir/message name>
processed messages: <# messages sucessfully dequeued>
failed processing attempts: <#messages reenqueued>
<count of messages> MESSAGES BEING PROCESSES BY JOB <pid>
<count of jobs> JOBS ACTIVE FOR CHANNEL foo
<count of active channels> ACTIVE CHANNELS
```

quit

Exits the `imsimta qm` utility. Synonymous with the `exit` sub-command.

Available in both interactive and non-interactive modes.

```
quit
```

read

Displays the specified messages from the most recently generated message queue listing.

Available only in interactive mode.

```
read [-content | -nocontent ] [-channel=<name> [-all]]
[-confirm | -noconfirm] [<id>...]
```

The options for this sub-command are:

Option	Description
`-content` | `-nocontent`	Displays (`-content`) or suppresses display (`-noconten}}t) of message content along with the envelope and header information. {{-nocontent` is the default.
id	Specifies the messages to display.

See imsimta qm Options for information on using the `-channel`, `-all`, and `-confirm` options.

release

If the specified message file is marked as held, it is renamed to remove the hold mark. The Job

Controller, if running, is informed that the message is to be processed immediately, ahead of all other messages.

Available only in interactive mode.

```
release [-channel=<name> [-all]] [-confirm | -noconfirm]
[-log | -nolog] [<id>...]
```

Use the *id* parameter to specify the messages to release from `.HELD` status.

Note
Run the `dir` subcommand prior to running `release`. For example:

```
$ imsimta qm
qm maint> dir -held
qm maint> release 1
```

See imsimta qm Options for information on using the `-channel`, `-all`, `-confirm`, and `-log` options.

return

Returns as undelivered the specified messages shown in the most recently generated message queue listing.

Available only in interactive mode.

```
return [-channel=<name> [-all]] [-confirm | -noconfirm]
[-log | -nolog] [<id>...]
```

Use the *id* parameter to specify the messages to return.

See imsimta qm Options for information on using the `-channel`, `-all`, `-confirm`, and `-log` options.

run

Processes, line-by-line, the commands specified in a file.

Available in both interactive and non-interactive modes.

```
run [-ignore | -noignore] [-log | -nolog] <file-spec>
```

Specifically, *file-spec* is opened and each line from it is read and executed.

The options for this sub-command are:

Option	Description
`-ignore` | `-noignore`	Unless `-ignore` is specified, command execution will be aborted should one of the sub-commands generate an error.
`-log` | `-nolog`	By default, each command is echoed to the terminal before being executed (the `-log` option). Specify `-nolog` to suppress this echo.

start

Restart processing of messages enqueued for the specified channel. The Job Controller not only marks the channel as "okay" to process, but it additionally starts processing jobs for the channel. This command takes effect whether the Job Controller is running or not.

```
start <channel>
```

The *channel* parameter specifies the channel to restart.

Note
The command `imsimta qm start/stop }}` *`channel`* `might fail if run simultaneously for many channels at the same time. The tool might have trouble updating the hold_list and could report: "QM-E-NOTSTOPPED, unable to stop the channel; cannot update the hold list." {{imsimta qm start/stop` *channel* should only be used sequentially with a few seconds interval between each run. If you only want the channel to run between certain hours, use the following options in the channel definition section in the job controller configuration file:

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

stop

Stops processing of messages enqueued for the specified channel. This command prevents you from having to stop the Job Controller and recompiling the configuration. The channel does not process messages until a `start` command is issued for that channel. This state persists across restarts of the Job Controller, the Messaging Server, and the host computer itself. This command takes effect whether the Job Controller is running or not.

```
stop <channel>
```

The *channel* parameter specifies the channel to stop.

Note
The command {{imsimta qm start/stop }} *channel* might fail if run simultaneously for many channels at the same time. The tool might have trouble updating the hold_list and could report: "QM-E-NOTSTOPPED, unable to stop the channel; cannot update the hold list." {{imsimta qm start/stop }} *channel* should only be used sequentially with a few seconds interval between each run. If you only want the channel to run between certain hours, use the following options in the channel definition section in the job controller configuration file:

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

summarize

The `summarize` sub-command displays a summary listing of message files.

```
summarize [-database | -directory_tree] [-heading | -noheading]
[-held | -noheld] [-trailing | -notrailing]
```

The options for this sub-command are:

Option	Description
`-database` | `-directory_tree`	Controls whether the information presented is obtained from the Job Controller (`-database`) or by looking at the actual directory tree containing the channel queues (`-directory_tree`). When neither `-database` nor `-directory_tree` is specified, then the "view" selected with the `view` sub-command will be used. If no `view` sub-command has been issued, then `-directory_tree` is assumed.
`-heading` | `-noheading`	Controls whether or not a heading line describing each column of output is displayed at the start of the summary listing. The {{-headin}}g option is the default.
`-held` | `-noheld`	Controls whether or not to include counts of .HELD messages in the output. The `-noheld` option is the default.
`-trailing` | `-notrailing`	Controls whether or not a trailing line with totals is displayed at the end of the summary. The `-trailing` option is the default.

top

The top sub-command displays the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. When used in conjunction with the `clean` sub-command, `top` may be used to locate unsolicited bulk email in the query and hold or delete it.

```
top [-content[=<range>]] [-from[=<range>]] [-subject[=<range>]]
[-to[=<range>]] [-database | -directory_tree] [-domain_to[=<range>]]
[-held] [-ignore_zz] [-min_count=<n>] [-threads=<n> | -nothreads]
[-top=<n>] [-verbose | -noverbose] [<channel>]
```

The options for this sub-command are:

Option	**Description**
`-content`[=*range*] `-from`[=*range*] `-subject`[=*range*] `-to`[=*range*] `-domain_to`[=*range*]	The `-content`, `-from`, `-subject`, and `-to` options are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Use `-from` to display frequent envelope From: fields, `-to` to display frequent envelope To: fields, or `-content` to display frequent message contents. Use `-domain_to` to display frequently occurring envelope To: addresses. Identical to `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.Any combination of `-content`, `-from`, `-to`, `-domain_to`, and `-subject` may be specified. However, only one of each may be used. The `-content`, `-from`, `-to`, `-domain_to`, and `-subject` options accept the optional parameters `START=`*n* and `LENGTH=`*n*. These parameters indicate the starting position and number of bytes in the field to consider. The defaults are `-content=(START=1,LENGTH=256)`, `-from=(START=1,LENGTH=2147483647)`, `-to=(START=1,LENGTH=2147483647)`, `-subject=(START=1,LENGTH=2147483647)`, and `-domain_to=(START=1,LENGTH=214783647)`. Use of these parameters is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.
`-database` | `-directory_tree`	Controls whether the message files scanned are only those with entries in the queue cache database (`-database`) or all message files actually present in the channel queue directory tree (`-directory_tree`). When neither `-database` nor `-directory_tree` is specified, then the "view" selected with the `view` sub-command will be used. If no `view` sub-command has been issued, then `-directory_tree` is assumed.
`-held`	Lists only the files which have a `.HELD` extension.
`-ignore_zz`	Ignores queued message files with file names beginning with "ZZ." This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt.
`-min_count=`*n*	Changes the minimum number of times that a string must occur in order to be displayed. The default is `-min_count=2`.
`-threads=`*n* | `-nothreads`	Accelerates 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 between `1` and `8`. The default is `-nothreads`.
`-top=`*n*	Changes the amount of most frequently occurring fields that are displayed. The default is `-top=20`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies an MTA channel area to be scanned for string frequencies. The `*` or? wildcard characters may be used in the channel specification.

view

Specifies the source of queued message information for subsequent directory commands.

Available only in interactive mode.

```
view -database | -directory_tree
```

By default, queued message listings are generated by scanning the `imta/queue/` directory tree. This corresponds to the `-directory_tree` option. You can, alternatively, generate the listings from the MTA queue cache database by issuing the `-database` option.

Settings made with the `view` sub-command remain the default until either another view command is issued or the utility exits. The default may be overridden with the `-database` or `-directory_tree` options of the directory command.

Note that the directory tree is always used when listing held message files.

version

Diagnostic command which outputs the Messaging Server version information as well as the compiled date and time for the `qm` program being run.

Available only in interactive mode.

```
version
```

imsimta qm Options

The `delete`, `history`, `hold`, `read`, `release`, and `return` sub-commands all support the following options and parameter:

Option	Description
`-channel=` *name*	Operates on the specified channel.
`-all`	The `-all` option may be used to operate on all of the previously listed messages. When used in conjunction with the -channel option, only those previously listed messages for the specified channel are operated on. The -all option may not be used in conjunction with an id parameter. However, -all or at least one id parameter must be specified.
`-confirm` and `-noconfirm`	`-confirm` prompts you to confirm each message release operation when the *id* parameter is not used to explicitly select messages. This prevents accidental `delete -all` sub-commands from being executed. The `-noconfirm` option suppresses this prompt.
`-log` and `-nolog`	Controls whether or not the operation on each selected message is reported.
id	The identification number of a message shown in the most recent listing generated by either the `directory` or the `held` sub-command. The identification number for a message is the integer value displayed in the left-most column of the listing. The *id* can also be a range or comma-separated list.

These options identify the messages to which the command is applied. When none of the options are specified, at least one *id* parameter must be supplied.

For example, in the following listing the first message displayed has an identification number of 1 and the second 2:

```
qm.maint> directory tcp_local
Sat, 26 Dec 2009 13:31:27 +1100 (EST)
Data gathered from the queue directory tree

Channel: tcp_local Size Queued since
--------------------------------------------------------------
1 ZZg0N2j0Pzpq0.00 1 26 Dec 2009 13:31:08
2 ZZg0N2j0Pzqq1.00 1 26 Dec 2009 13:31:12
--------------------------------------------------------------
Total size: 2

Grand total size: 2
```

These two messages can therefore be selected by either "1,2" or "1-2".

Examples

Non-Interactive Mode

The following example generates a list of queued messages:

```
bash-3.00# ./imsimta qm directory
Sat, 26 Dec 2009 13:32:26 +1100 (EST)
Data gathered from the queue directory tree

Channel: ims-ms Size Queued since
--------------------------------------------------------------
1 ZZg0N2j0PzEq2.00 1 26 Dec 2009 13:32:08
2 ZZg0N2j0PzHq3.00 1 26 Dec 2009 13:32:20
--------------------------------------------------------------
Total size: 2


Channel: tcp_local Size Queued since
--------------------------------------------------------------
3 ZZg0N2j0Pzpq0.00 1 26 Dec 2009 13:31:08
--------------------------------------------------------------
Total size: 1

Grand total size: 3
```

Interactive Mode

In the following interactive session, the `directory` sub-command is used to obtain a list of queued messages. The `delete` sub-command is then used to delete the first of the displayed messages. Finally, another `directory` sub-command is issued that displays that the deleted message is indeed gone.

```
bash-3.00# ./imsimta qm
qm.maint> directory
Sat, 26 Dec 2009 13:32:47 +1100 (EST)
Data gathered from the queue directory tree

Channel: ims-ms Size Queued since
--------------------------------------------------------------
1 ZZg0N2j0PzEq2.00 1 26 Dec 2009 13:32:08
2 ZZg0N2j0PzHq3.00 1 26 Dec 2009 13:32:20
--------------------------------------------------------------
Total size: 2


Channel: tcp_local Size Queued since
--------------------------------------------------------------
3 ZZg0N2j0Pzpq0.00 1 26 Dec 2009 13:31:08
--------------------------------------------------------------
Total size: 1

Grand total size: 3
qm.maint> delete 1
%QM-I-DELETED, deleted the message file
/opt/sun/comms/messaging/data/queue/ims-ms/000/ZZg0N2j0PzEq2.00
qm.maint> directory
Sat, 26 Dec 2009 13:32:53 +1100 (EST)
Data gathered from the queue directory tree

Channel: ims-ms Size Queued since
--------------------------------------------------------------
1 ZZg0N2j0PzHq3.00 1 26 Dec 2009 13:32:20
--------------------------------------------------------------
Total size: 1


Channel: tcp_local Size Queued since
--------------------------------------------------------------
2 ZZg0N2j0Pzpq0.00 1 26 Dec 2009 13:31:08
--------------------------------------------------------------
Total size: 1

Grand total size: 2
```

imsimta qtop

The `imsimta qtop` utility displays the most frequently occurring envelope From:, To:, Subject:, or message content fields found in message files in the channel queues.

Syntax

```
imsimta qtop [-content[=<range>]] [-from[=<range>]] [-subject[=<range>]]
[-to[=<range>]] [-domain_to[=<range>]] [-database | -directory_tree]
[-ignore_zz] [-min_count=<n>] [-threads=<n> | -nothreads] [-top=<n>]
[-verbose | -noverbose] [<channel>]
```

Options

The options for this command are:

Option	Description
`-content[=`*range]* `-from[=`*range]* `-subject[=`*range]* `-to[=`*range]* `-domain_to[=`*range]*	Specifies which frequently occurring fields should be displayed. By default, only Subject: fields are shown (`-subject`). Specify `-from` to display frequent envelope From: fields, `-to` to display frequent envelope To: fields, or `-content` to display frequent message contents. Specify `-domain_to` to display frequently occurring envelope To: fields. Identical to `-to` option, except `-domain_to` looks at only the *host.domain* portion of the envelope To: address.Any combination may be specified. However, only one of each my be used. These options accept the `START=`*n* and `LENGTH=`*n* arguments. These arguments indicate the starting offset and number of bytes in the field to consider. The defaults are `-content=(START=1,LENGTH=256)`, `-from=(START=1,LENGTH=2147483647)`, `-subject=(START=1,LENGTH=2147483647)`, and `-domain_to=(START=1,LENGTH=2147483647)`.
`-database`	Specifies that only message files identified by the queue cache database is searched.
`-directory_tree`	Searches all message files actually present in the channel queue directory tree.
`-ignore_zz`	Ignores queued message files with file name beginning with "ZZ." This option may be used to scan only those message files which represent queued messages which have failed at least one delivery attempt. For example, the following command indicates to which domains the MTA has problems delivering messages: `imsimta qtop -ignore_zz -domain_to`
`-min_count=`*n*	Changes the minimum number of times that a string must occur in order to be displayed. The default is `-min_count=2`.
`-threads=`*n* | `-nothreads`	Accelerates 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 between `1` and `8`. The default is `-nothreads`.
`-top=`*n*	Changes the amount of most frequently occurring fields that are displayed. The default is `-top=20`.
`-verbose` | `-noverbose`	Requests that the utility displays operation information (`-verbose`). The default is `-noverbose`.
channel	Specifies a channel area to be scanned for string frequencies. The `*` and `?` wildcard characters may be used in the channel specification.

imsimta refresh

The `imsimta refresh` utility performs the following functions:

- Recompiles the MTA configuration files.
- Stops any MTA Job Controller or MTA Service Dispatcher jobs that are currently running.
- Restarts the Job Controller and MTA Service Dispatcher.

Essentially, `imsimta refresh` combines the function of `imsimta cnbuild` and `imsimta restart`.

Note
You must be logged in as `root` to run `imsimta refresh`.

Note
This command has been deprecated. On a production system, the imsimta refresh command should only be used as a last resort. This is because the command does far more than refresh the running MTA services. Specifically, it shuts down all running MTA services, builds a new compiled configuration from the inactive, human readable configuration files, and then restarts all the MTA services with the new compiled configuration. In shutting down the MTA services, all heuristic queuing information is lost (e.g., redelivery schedules). More often than not, the command `imsimta restart` with the specific MTA service which needs to be restarted (for example, `smtp`, `dispatcher`, `job_controller`). When a configuration change has been made and an MTA service needs to pick up that change, then first use the `imsimta cnbuild` followed by the `imsimta restart` command. Also, strongly consider using the `imsimta reload` command instead of the `imsimta restart` command.

Syntax

```
imsimta refresh [job_controller | dispatcher]
```

Options

The options for this command are:

Option	Description
`job_controller`	Restarts the Job Controller.
`dispatcher`	Restarts the MTA Service Dispatcher.

If no component name is specified, all active components are restarted.

imsimta reload

Some parts of the MTA configuration can be changed and have these changes activated without having to stop and start the system. The reloadable parts of the configuration are:

mappings

aliases

general, *forward* and *reverse* lookup tables

These can be changed, compiled, and the changes activated by issuing the commands:

`imsimta cnbuild`

```
imsimta reload
```

The imsimta reload command informs the dispatcher and job controller of the change, and they in turn inform the processes they started.

Syntax

```
imsimta reload
```

imsimta renamedb

The `imsimta renamedb` command renames an MTA database. Since the MTA may optionally reference several "live" databases, that is, databases whose presence triggers their use by the MTA, it is important, first, to ensure that the MTA does not see such a database while it is in a mixed state, and second, to minimize any period of time during which the database is inaccessible. The `imsimta crdb` command locks the database it is creating to avoid having it accessed in a mixed state.

It is therefore recommended that the MTA databases be created or updated in a two-step process:

1. Create or update a temporary database.
2. Rename the temporary database to the "live" name by using the `imsimta renamedb` command.

The `imsimta renamedb` command, which must delete any old database files and rename the new database files, locks the database during the renaming process to avoid presenting the database in a mixed state. In this way the database is never accessible while it is in a mixed state, yet any window of time during which the database is inaccessible is minimized. Renaming is generally quicker than database generation.

Syntax

```
imsimta renamedb <old-database-spec> <new-database-spec>
```

Parameters

The parameters for this command are:

Parameter	Description
old-database-spec	The name of the database that is being renamed.
new-database-spec	The new name of the database. This may either be an actual pathname, or one of the special names such as `IMTA_ALIAS_DATABASE`, `IMTA_REVERSE_DATABASE`, `IMTA_GENERAL_DATABASE`, or `IMTA_DOMAIN_DATABASE`, listed in the MTA tailor file and pointing to actual pathnames.

Example

The following command renames the database `tmpdb` to be the actual MTA alias database (usually *msg-svr-base*`/data/db/aliasesdb`).

```
imsimta renamedb tmpdb IMTA_ALIAS_DATABASE
```

imsimta restart

The `imsimta restart` command stops and restarts the Job Controller and Service Dispatcher. This causes all MTA master and slave programs to be restarted. It can also restart SMTP, LMTP, and SMTP_SUBMIT.

Detached MTA processes should be restarted whenever the MTA configuration is altered. These processes load information from the configuration only once and need to be restarted in order for configuration changes to become visible to them. In addition to general MTA configuration files, such as the `imta.cnf` file, some components, such as the MTA Service Dispatcher, have their own specific configuration files, for example, `dispatcher.cnf,` and should be restarted after changes to any of these files.

Note
You must be logged in as `root` to use this utility.

Syntax

```
imsimta restart [job_controller|dispatcher|smtp|lmtp|smtp_submit]
```

Restarting the MTA Service Dispatcher effectively restarts all the service components it handles. If no component name is given, all active components are restarted.

Example

To restart the MTA Job Controller and channel master programs:

```
imsimta restart job_controller
```

imsimta return

The `imsimta return` command returns a message to the message's originator. The returned message a single multipart message with two parts. The first part explains the reason why the message is being returned. The text of the reason is contained in the file `return_bounce.txt` located in the *msg-svr-base*`/config/locale/C/LC_MESSAGES` directory. The second part of the returned message contains the original message.

Syntax

```
imsimta return <message-file>
```

message-file is the name of the message file to return. The name may include wildcards, but if so, the specification must be quoted.

Example

The following command causes the specified the message to be returned to its originators.

```
imsimta return /imta/queue/l/ZZ0FRW00A03G2EUS.00
```

imsimta run

The `imsimta run` command processes the messages in the channel specified by the channel parameter. Output during processing is displayed at your terminal, which makes your terminal unavailable for the duration of the operation of the utility. Refer also to the `imsimta submit` command which, unlike `imsimta run`, does not monopolize your terminal.

Note that a channel delivery program that is run using this command, unlike the `imsimta submit` command, attempts to deliver messages before any pending backoff delay has expired.

Syntax

```
imsimta run <channel>
```

Parameters

The parameter for this command is:

Parameter	Description
channel	Specifies the channel to be processed. This parameter is mandatory.

Example

Type the following command to process any messages in the tcp_local channel:

```
imsimta run tcp_local
```

imsimta shutdown

The `imsimta shutdown` command shuts down the MTA Job Controller and the MTA Dispatcher. Shutting down the MTA Dispatcher shuts down all services (for example, SMTP) being handled by the Dispatcher. It can also be used to stop the SMTP, LMTP, SMTP_SUBMIT servers. Note that you can only restart a Dispatcher service that is currently running. If you do `imsimta shutdown smtp`, you must restart the Dispatcher to start the SMTP service again.

Note
You must be logged in as `root` to use this utility.

Syntax

```
imsimta shutdown [dispatcher|job_controller|smtp|smtp_submit|lmtp]
```

Example

Use the following command to shut down the MTA jobs:

```
imsimta shutdown
```

imsimta start

The `imsimta start` command starts up detached MTA processes. If no component parameter is specified, then the MTA Job Controller and MTA Service Dispatcher are started. Starting the Service Dispatcher starts all services the Service Dispatcher is configured to handle, which usually includes the SMTP server.

The services handled by the MTA Service Dispatcher must be started by starting the MTA Service Dispatcher. Only services not being handled by the MTA Service Dispatcher can be individually started via the `imsimta start` command. The Service Dispatcher may be configured to handle various services, for example, the multithreaded SMTP server.

> **Note**
> You must be logged in as `root` to use this utility.

Syntax

```
imsimta start [<component>]
```

If a component parameter is specified, then only detached processes associated with that component are started. The standard component names are:

- `dispatcher` – Multithreaded Service Dispatcher.
- `job_controller` – Schedules deliveries (dequeues messages).

Example

Use the following command to start the MTA Job Controller and MTA Service Dispatcher:

```
imsimta start
```

imsimta stop

The `imsimta stop` command shuts down the MTA Job Controller and the MTA Dispatcher. Shutting down the MTA Dispatcher shuts down all services (for example, SMTP) being handled by the Dispatcher. It can also be used to stop the SMTP, LMTP, SMTP_SUBMIT servers. Note that you can only restart a Dispatcher service that is currently running. If you do `imsimta shutdown smtp`, you must restart the Dispatcher to start the SMTP service again.

Note
You must be logged in as `root` to use this utility.

Syntax

```
imsimta stop [dispatcher|job_controller|smtp|smtp_submit|lmtp]
```

Example

Use the following command to shut down the MTA jobs:

```
imsimta stop
```

imsimta submit

The `imsimta submit` command directs the Job Controller to fork a process to execute the messages queued to the channel specified by the channel parameter.

Syntax

```
imsimta submit [<channel>] [poll]
```

Parameters

The parameters for this command are:

Parameter	Description
channel	Specifies the channel to be processed. The default, if this parameter is not specified, is the local channel `1`.
`poll`	If poll is specified, the channel program runs even if there are no messages queued to the channel for processing.

Example

Use the following command to process any messages in the `tcp_local` channel:

```
imsimta submit tcp_local
```

imsimta test

The `imsimta test` utilities perform tests on various areas of functionality of the MTA.

The `imsimta test -domain` utility is an interactive command with sub-commands that differ from the options common to the other `imsimta test` utilities. For information about `imsimta test -domain`, see imsimta test -domain.

imsimta test -mapping

`imsimta test -mapping` tests the behavior of a mapping table in the mapping file. The result of mapping an input string will be output along with information about any meta characters 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, `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 is entered.

imsimta test -match

`imsimta test -match` tests a mapping pattern in order to test wildcard and global matching.

`imsimta test -match` prompts for a pattern and then for a target string to compare against the pattern. The output indicates whether or not the target string matched. If a match was made, the characters in the target string that matched each wildcard of the pattern is displayed. The `imsimta test -match` utility loops, prompting for input until the utility is exited with a CTRL-D.

imsimta test -rewrite

`imsimta test -rewrite` provides a test facility for examining the MTA's address rewriting and channel mapping process without actually sending a message. Various qualifiers can be used to control whether `imsimta test -rewrite` uses the configuration text files or the compiled configuration (if present), the amount of output produced, and so on.

If a test address is specified on the command line, `imsimta test -rewrite` applies the MTA address rewriting to that address, reports the results, and exits. If no test address is specified, `imsimta test -rewrite` enters a loop, prompting for an address, rewriting it, and prompting again for another address. `imsimta test -rewrite` exits when CTRL-D is entered.

When testing an email address corresponding to a restricted distribution list, `imsimta test -rewrite` uses as the posting address the return address of the local postmaster, which is usually `postmaster@localhost` unless specified by the MTA option RETURN_ADDRESS in the MTA Option file.

imsimta test -url

`imsimta test -url` tests an LDAP queury URL. Note that the LDAP server to query is controlled by the setting of the MTA option `LDAP_SERVER` in `local.conf`.

Syntax

```
imsimta test -rewrite [-alias_file=<filename>]
[-channel | -nochannel
[-check_expansions | -nocheck_expansions]
[-configuration_file=<filename> ] [-database=<database_list>]
[-debug | -nodebug] [-delivery_receipt | -nodelivery_receipt]
[-destination_channel=<channel>] [-expandlimit=<integer>]
[-extra_local_channel=<channel>] [-filter | -nofilter]
[-from=_address_ | -nofrom] [-image_file=<filename> | -noimage_file]
[-input=<input-file>] [-local_alias=<value> | -nolocal_alias]
[-mapping_file=<file> | -nomapping_file]
[-option_file=<filename> | -nooption_file] [-output=<output-file>]
[-read_receipt | -noread_receipt] [-noreprocess] [-restricted=<setting>]
[-sender=_from_address_] [-source_channel=<channel>]
[-soptin] [-spares]
```

```
imsimta test -mapping [<input_string>] [-debug | -nodebug]
[-flags=<chars> | -noflags]
[-image_file=<filename> | -noimage_file] [-mapping_file=<filename>]
[-option_file=<filename> | -nooption_file] [-table=<table-name>] [<address>]
```

```
imsimta test -match
```

```
imsimta test -url [-debug | -nodebug] [<ldap_url>]
```

```
imsimta test -exp -mm -message=<message-file>[-block] [-input=<input-file>]
[-output=<output-file>] [-sender=<authenticated-sender>]
[-username=<username>]
```

Options

The options for this command are:

Option	Description
address	Specifies the test address to be rewritten. If this option is omitted, then the command prompts for an address. Used with the `-rewrite` option.
input_string	The string to be matched in the left side of a mapping table. Used with the `-mapping` option.
ldap_url	The LDAP URL that `imsimta test -url` attempts to resolve.

`-alias_file=`*filename*	Specifies an alternate alias file for `imsimta test -rewrite` to use. `imsimta test -rewrite` normally consults the default alias file named by the `IMTA_ALIAS_FILE` option of the MTA tailor file, *msg-svr-base* `/config/imta_tailor`, during the rewriting process. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; any compiled configuration precludes reading any sort of alias file.
`-block`	Treats the entire input as a single sieve script. The default is to treat each line as a separate script.
`-channel` | `-nochannel`	Controls whether `imsimta test -rewrite` outputs detailed information regarding the channel an address matches (for example, channel flags).
`-check_expansions` | `-nocheck_expansions`	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` option 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.
`-configuration_file=` *file*	Specifies an alternate file to use in place of the file named by `IMTA_CONFIG_FILE`. Normally, `imsimta test -rewrite` consults the default configuration file named by the `IMTA_CONFIG_FILE` option of the MTA tailor file, *msg-svr-base*`/config/imta_tailor`, during the rewriting process. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of configuration file.
`-database=`*database-list*	Disables references to various databases or redirects the database paths to nonstandard locations. `imsimta test -rewrite` normally consults the usual MTA databases during its operation. The allowed list items are `alias`, `noalias`, `domain`, `nodomain`, `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.
`-debug` | `-nodebug`	Enables the production of the additional, detailed explanations of the rewriting process. This option is disabled by default.
`-delivery_receipt` | `-nodelivery_receipt`	Sets the corresponding receipt request flags. These options can be useful when testing the handling of sent or received receipt requests when rewriting forwarded addresses or mailing lists.
`-destination_channel` =*channel*	Controls to which destination or target channel `imsimta test -rewrite` rewrites addresses. Some address rewriting is destination channel specific; `imsimta test -rewrite` normally pretends that its channel destination is the local channel l.
`-envid`	Specifies the envelope identifier. For example, a@b.

`-exp`	`imsimta test -exp` tests Sieve language statements against a specified RFC2822 message and sends the results of the filter to standard output.The syntax is as follows: imsimta test -exp -mm -block -input=Sieve_language_scriptfile -message=rfc2822_message_filewhere, -block treats the entire input as a single Sieve script. The default is to treat each line as a separate script and to evaluate it separately. The Sieve will only be evaluated once the end of file is reached. -input=Sieve_file is a file containing the Sieve script. The default is to read the test script lines or script block from stdin. -message=message_file is a text file containing the RFC 2822 message you want to test your Sieve script against. This has to be an RFC 2822 message only. It cannot be a queue file (not a zz*.00 file).Once activated, this command reads script information, evaluates it in the context of the test message, and writes out the result. The result shows what actions would be taken as well as the result of evaluating the final statement in the script.Additional useful qualifiers are: -from=address specifies the envelope from address to be used in envelope tests. The default is to use the value specified by the RETURN_ADDRESS MTA option. -output=file writes results to file. The default is to write the results of script evaluation to stdout. -sender=<authenticated-sender> sets the authenticated sender address. -username=<username> sets the authentication username. Note that -sender and -username qualifiers simply simulate the setting of these fields. They do not actually perform any sort of authentication operation.
`-expandlimit=`*integer*	The integer value is used to set the internal alias expansion limit within the MTA. The switch is intended to be used for testing expansion limit interactions with mailing lists and other MTA facilities.
`-extra_local_channel` =*channel*	(New in MS7u2) Tell the rewriting machinery that "channel" should have local channel semantics. This is useful in testing rewriting associated with process and conversion channel hops.
`-filter` | `-nofilter`	Outputs any filters that are applied for the specified address.
`-from=`*address* | `-nofrom`	Controls what envelope From: address is used for access control probes when the `-from` option is specified. If *address* is omitted, the postmaster return address is used for such probes. If the `-nofrom` option is specified, the MTA uses an empty envelope From: address for access probes.
`-flags=`*chars* | `-noflags`	Specifies particular flags to be set during the mapping test when the `-flags` option is specified. For example, *chars* can be E (envelope), B (header/body), or I (message id) when testing a REVERSE mapping. Used with the `-mapping` option only.
`-image_file=`[*filename*] | `-noimage_file`	The `-noimage_file` option instructs the command to unconditionally ignore any previously compiled configuration and to read configuration information from the various text files instead. When the `-image_file` option is specified without an optional file name, the compiled configuration is loaded from the file named by the `IMTA_CONFIG_DATA` option into the MTA tailor file, *msg-svr-base*`/config/imta_tailor`, which is usually *msg-svr-base*`/config/imta.cnf`. If, instead, a file name is specified, then the compiled configuration is loaded from the specified file.
`-input=`*input-file*	Specifies a source for input. By default, `imsimta test` takes input from `stdin`.

`-local_alias=`*value* | `-nolocal_alias`	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 option may be used to set the local host alias to the specified value; appearances of the local host in rewritten addresses are replaced by this value.
`-mapping_file=`*file* | `-nomapping_file`	Instructs the command to use the specified mapping file rather than the default mapping file named by the `IMTA_MAPPING_FILE` option in the MTA tailor file, _msg-svr-base_`/config/imta_tailor`, which is usually the file named _msg-svr-base_`/config/mappings`. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of any compiled configuration precludes reading the mappings file. Use of the `-nomapping_file` option will prevent the `IMTA_MAPPING_FILE` file from being read in when there is no compiled configuration.
`-message=`*message-file*	Specifies the text file containing the message that is tested. The *message-file* must be an RFC 822 message only; it cannot be a queue file.
`-mm`	Tells `imsimta test -exp` to load the sieve-specific extensions to the expression interpreter. This includes all the sieve tests and actions such as `header`, `address`, `envelope`, `discard`, `fileinto`, and `keep`. Without `-mm` you cannot test sieves. The command to test sieves against a message is:`imsimta test -expression -mm -message=` *message*
`-noreprocess`	Turns off the internal reprocessing flag that would otherwise be set. This option is useful for simulating the behavior of other components that operate without the reprocessing flag being set. This can be thought of as controlling whether or not `rewrite_test` acts as if it were the reprocessing channel. The biggest effect is that it turns off deferred list processing. Normally it should be done so this switch defaults on; use `-noreprocessing` to disable expansion.
`-option_file=`*filename* | `-nooption_file`	Instructs the command to use the specified option file rather than the default option file named by the `IMTA_OPTION_FILE` option in the MTA tailor file, *msg-svr-base*`/config/imta_tailor`, which is usually the file *msg-svr-base*`/config/options.dat`. This option has no effect unless `-noimage_file` is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of option file. Use of the `-nooption_file` option prevents the `IMTA_OPTION_FILE` file from being read in when there is no compiled configuration.
`-output=`*output-file*	Directs the output of `imsimta test`. By default, `imsimta test` writes output to `stdout`. This option only works if the `mailsrv` account has write access to the current working directory.
`-read_receipt` | `-noread_receipt`	Sets the corresponding receipt request flags. This option can be useful when testing the handling of receipt requests at the time of rewriting forwarded addresses or mailing lists.
`-restricted=`*setting*	Controls the setting of the restricted flag. By default, this flag has value `0`. When set to `1`, `-restricted=1`, the restricted flag is set on and addresses are 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.

`-sender=`*from_address*	A value used to set the "authenticated sender" (final field) of `FROM_ACCESS` mapping table probes. That is, one received as a result of SASL authentication. This allows `test -rewrite` to be used to test these mappings.
`-soptin`	(New in MS7u2) Show any optins associated with the recipient address.
`-source_channel=` *channel*	Controls which source channel is performing the rewriting. Some address rewriting is source channel-specific; `imsimta test -rewrite` normally assumes that the channel source for which it is rewriting is the local channel `l`.
`-spares`	(New in MS7u2) Show any spare attributes associated with the recipient address.
`-table=`*table-name*	Specifies the name of the mapping table to test. If this option is not specified, then `imsimta test -mapping` prompts for the name of the table to use.

Example

This example shows typical output generated by `imsimta test -rewrite`. The most important piece of information generated by `imsimta test -rewrite` is displayed on the last few lines of the output, which shows the channel to which `imsimta test -rewrite` would submit a message with the specified test address and the form in which the test address would be rewritten for that channel. This output is invaluable when debugging configuration problems.

```
imsimta test -rewrite

Address: joe.blue
channel = l
channel description =
channel description =
channel flags #1 = BIDIRECTIONAL MULTIPLE IMMNONURGENT NOSERVICEALL
channel flags #2 = NOSMTP POSTHEADBODY HEADERINC NOEXPROUTE
channel flags #3 = LOGGING NOGREY NORESTRICTED
channel flags #4 = EIGHTNEGOTIATE NOHEADERTRIM NOHEADERREAD RULES
channel flags #5 =
channel flags #6 = LOCALUSER NOX_ENV_TO RECEIPTHEADER
channel flags #7 = ALLOWSWITCHCHANNEL NOREMOTEHOST DATEFOUR DAYOFWEEK
channel flags #8 = NODEFRAGMENT EXQUOTA REVERSE NOCONVERT_OCTET_STREAM
channel flags #9 = NOTHURMAN INTERPRETENCODING

text/plain charset def = (7) US-ASCII 5 (8) ISO-8859-1 51
channel envelope address type = SOURCEROUTE
channel header address type = SOURCEROUTE
channel official host = mailserver.eng.alpha.com
channel local alias =
channel queue name =
channel after param =
channel daemon name =
channel user name =
notices =
channel group ids =
header To: address = joe.blue@mailserver.eng.alpha.com
header From: address = joe.blue@mailserver.eng.alpha.com
envelope To: address = joe.blue@mailserver.eng.alpha.com
(route (mailserver.eng.alpha.com,mailserver.eng.alpha.com))
envelope From: address = joe.blue@mailserver.eng.alpha.com
name =
mbox = joe.blue
Extracted address action list: joe.blue@mailserver.eng.alpha.com
Extracted 733 address action list: joe.blue@mailserver.eng.alpha.com
Expanded address:
joe.blue@mailserver.eng.alpha.com
Submitted address list:
ims-ms
joe.blue@ims-ms-daemon (sims-ms-daemon) *NOTIFY FAILURES* *NOTIFY DELAYS*
Submitted notifications list:
Address:
#
```

In the following example, the sample PAGER mapping is tested. The `-mapping_file` option is used to select the mapping file `pager_table.sample` instead of the default mapping file.

```
imsimta test -mapping -noimage_file \
-mapping_file=<msg-svr-base>/config/pager_table.sample
```

In the following example, the sample mapping pattern `$[ax1]@.xyz.com` is tested for several sample target strings:

```
imsimta test -match

Pattern: $[ax1]*@*.xyz.com
[ 1S] cglob [1ax]
[ 2] "@"
[ 3S] glob, req 46, reps 2
[ 4] "."
[ 5] "x"
[ 6] "y"
[ 7] "z"
[ 8] "."
[ 9] "c"
[ 10] "o"
[ 11] "m"
Target: xx11aa@sys1.xyz.com
Match.
0 - xx11aa
1 - sys1
Pattern: $[ax1]*@*.xyz.com
Target: 12a@node.xyz.com
No match.
Pattern: $[ax1]*@*.xyz.com
Target: 1xa@node.acme.com
Match.
0 - 1xa
1 - node
Pattern: ^D
%
```

imsimta test -domain

The `imsimta test -domain` utility verifies the validity of domain structures in the LDAP directory, lists all domains in the directory, displays domains' locations (DNs), displays the values of domain attributes, and performs other domain-related tests.

You must be `root` or `mailsrv` to run `imsimta test -domain`.

Syntax

The `imsimta test -domain` utility must be run in interactive mode. To run the utility, type:

```
imsimta test -domain
```

The utility displays the following prompt:

```
DOMAIN_MAP>
```

At the `DOMAIN_MAP>` prompt, you can enter the commands shown in imsimta test -domain Commands.

To exit the interactive mode, enter `exit` or `quit`.

imsimta test -domain Commands

The commands for this utility are:

Command	Description
`enumerate`	Displays a list of all available domains in the LDAP directory.
`exit`	Exits the interactive-mode utility.
`help`	Displays help (usage) for the commands.Note: The help command displays usage for the `canonicalize` and `user` commands, which are used internally by Messaging Server and have no effect on domains in the directory. Do not use these commands.
`locate domain` *domain name*	Selects (binds to) the domain entry specified with its domain name (*domain name*). Once you select a domain entry, you can display information about the domain and its attributes with the `show` and `query` *attribute* commands.
`locate basedn` *baseDN name*	Selects (binds to) the domain entry specified with its base DN (*baseDN name*). Enclose the specified base DN in single quotes (') or double quotes (").Once you select a domain entry, you can display information about the domain and its attributes with the `show` and `query` *attribute* commands.
`query` *attribute*	Displays the name and value of the domain attribute specified with the variable *attribute*. The specified attribute must be present in the domain currently selected for display.
`quit`	Exits the interactive-mode utility.
`release`	Releases the currently selected domain entry. Once you release a domain entry, no information is displayed with the `show` and `query` *attribute* commands.
`show`	Displays information about the currently selected domain, including the domain name, canonical domain name, base DN, and domain DN.
`verify`	Verifies domain-level Schema 1 and 2 information in the directory.

Examples

To display a list of all available domains in the directory:

```
imsimta test -domain
DOMAIN_MAP> enumerate

siroe.com
varrius.com
sesta.com
```

To verify the domain structure of all available domains:

```
imsimta test -domain
DOMAIN_MAP> verify

%DMAP-E-CANONICAL, Overlapping domains 'sesta.com' and
'west.sesta.com' defined by entries 'o=sesta.com,o=rootsuffix'
and 'o=west.sesta.com,o=sesta.com,o=rootsuffix' have different
canonical domains 'sesta.com' and 'west.sesta.com'.
%DMAP-E-NODOMAINNAME, Domain entry with DN 'o=mycompany,o=rootsuffix'
does not have a domain name.
%DMAP-E-DOMAININVALID, Domain name 'domain_tst.com'
defined/referenced by domain entry with
DN 'o=domain_tst.com,o=rootsuffix' is syntactically invalid.
```

To select a domain entry for display by specifying the domain name:

```
imsimta test -domain
DOMAIN_MAP> locate domain siroe.com

entry located
```

To select a domain entry for display by specifying the base DN:

```
imsimta test -domain
DOMAIN_MAP> locate basedn "o=siroe.com,o=rootsuffix"

entry located
```

To display information about the domain entry:

```
imsimta test -domain
DOMAIN_MAP> show

Domain name: siroe.com
Canonical name: siroe.com
Lower case canonical name: siroe.com
Base DN: o=siroe.com,o=rootsuffix
Domain DN: o=siroe.com,o=rootsuffix
```

To show the value of the `sunPreferredDomain` attribute for the selected domain:

```
imsimta test -domain
DOMAIN_MAP> query sunPreferredDomain

Attribute value(s):
[0] "siroe.com"
```

To verify domain-level Schema 1 and 2 information in the directory:

```
imsimta test -domain
DOMAIN_MAP> verify

Various checks are done by this utility, but the most important by far
is verification of canonical domain settings for domains with overlapping
user entries.

The verification utility can return the following fatal errors:

%DMAP-F-CANTGETDN, Cannot obtain DN of domain entry, directory error
%DMAP-F-INTDEFERROR, Internal defined flag error on domain '%.*s', aborting
%DMAP-F-INTHASHERROR, Internal hash error, aborting
%DMAP-F-INTTREESTRUCTERROR, Internal tree structure error, aborting

These are all indicative of an internal error in the verification code
and should never occur.

The following domain errors can be reported:

%DMAP-E-ALIASTOOLONG, Domain alias '%s' in entry with DN '%s' is too long
%DMAP-E-BASEDNTOOLONG, Base DN pointer '%s' in entry for domain '%.*s' is too
long
%DMAP-E-CANONICAL, Overlapping domains '%.*s' and '%.*s' defined by entries
'%.*s' and '%.*s' have different canonical domains '%.*s'
and '%.*s'
%DMAP-E-CANONICALINVALID, Canonical domain '%.*s' defined/referenced by
domain entry with DN '%.*s' is syntactically
invalid
%DMAP-E-CANONICALTOOLONG, Canonical name '%s' in entry for domain '%.*s'
is too long
%DMAP-E-CANTCONVDCDN, Cannot convert DN '%s' in DC tree to domain name
%DMAP-E-CANTEXTALIAS, Empty alias pointer attribute in '%.*s' domain alias
entry
%DMAP-E-DOMAININVALID, Domain name '%.*s' defined/referenced by domain entry
with DN '%.*s' is syntactically invalid
%DMAP-E-DOMAINMULTDEF, Domain '%s' multiply defined by entries with DNs '%s'
and '%s'
%DMAP-E-DOMAINTOOLONG, Domain '%s' in entry with DN '%s' is too long
%DMAP-E-DOMAINUNDEF, Domain name '%.*s' referenced by domain entry with DN
'%.*s' never defined
%DMAP-E-EMPTYCANONICAL, Domain '%.*s' has an empty canonical name
%DMAP-E-INVALIDBASEDN, Base DN pointer '%.*s' in entry for domain '%.*s'
is not a valid DN
%DMAP-E-MULTICANONICAL, Multivalued canonical name in entry for domain
'%.*s', used value '%s' ignored '%s'
%DMAP-E-NOBASEDN, Domain '%.*s' has no base DN
%DMAP-E-EMPTYBASEDN, Domain '%.*s' has an empty base DN
%DMAP-E-NODOMAINNAME, Domain entry with DN '%s' does not have a domain
name

The following warnings can be reported:

%DMAP-W-DISALLLOWEDATTR, Domain '%.*s' has a disallowed attribute '%s'
with value '%s'
%DMAP-W-DNTOOLONG, Domain entry DN '%s' is too long
%DMAP-W-EMPAPPSTAT, Domain '%.*s' has an empty application status
```

```
%DMAP-W-EMPDISALLLOWED, Domain '%.*s' has an empty disallowed attribute
'%s'
%DMAP-W-EMPDOMSTAT, Domain '%.*s' has an empty domain status
%DMAP-W-EMPUIDSEP, Domain '%.*s' has an empty UID separator
%DMAP-W-INVALIDAPPSTAT, Application status '%s' for domain '%.*s' is
invalid
%DMAP-W-INVALIDDOMSTAT, Domain status '%s' for domain '%.*s' is invalid
%DMAP-W-INVALIDUIDSEP, UID separator '%s' for domain '%.*s' is invalid
%DMAP-W-MULTDOMAINNAMES, Domain entry with DN '%s' has multiple domain
names, used value '%s' ignored '%s'
%DMAP-W-MULTIAPPSTAT, Multivalued application status in entry for domain
'%.*s', used value '%s' ignored '%s'
%DMAP-W-MULTIBASEDN, Multivalued base DN pointer in entry for domain
'%.*s', used value '%s' ignored '%s'
%DMAP-W-MULTIDOMSTAT, Multivalued domain status in entry for domain
'%.*s', used value '%s' ignored '%s'
%DMAP-W-MULTIUIDSEP, Multivalued UID separator in entry for domain '%.*s',
used value '%s' ignored '%s'
%DMAP-W-MULTIVALIAS, Multivalued alias pointer in entry for domain alias
'%.*s', used value '%s' ignored '%s'
%DMAP-W-NOBASEDNNODE, Base DN pointer '%.*s' in entry for domain '%.*s'
doesn't point at anything
```

```
%DMAP-W-NODOMAINNAME, Domain entry with DN '%s' has a blank domain alias
%DMAP-W-NOENTRIES, No domain entries found, aborting
```

imsimta version

The `imsimta version` command prints out the MTA version number, and displays the system's name, operating system release number and version, and hardware type.

Syntax

```
imsimta version
```

Example

To check the version of MTA you are running, execute the following command:

```
% imsimta version
```

imsimta view

The `imsimta view` utility displays log files.

Syntax

```
imsimta view <file-pattern> [-f <offset-from-first>] [-l <offset-from-last>]
```

Options

The options for this command are:

Option	Description
`-f=` *offset-from-first*	Displays the specified version of the log file (starting from 0). For example, to find the earliest (oldest) version of the file, specify `-f=0`. By default, `imsimta view` finds the most recent version of the log file.
`-l=` *offset-from-last*	Displays the last version of the specified file. For example, to display the most recent (newest) version of the file, specify `-l=0`. By default, `imsimta view` finds the most recent version of the file.
file-pattern	Specifies a filename pattern to view.

Chapter 3. Messaging Multiplexor Configuration

Messaging Multiplexor Configuration

- For descriptions of the configurable multiplexor parameters, see MMP Reference.
- For multiplexor tasks, see the topic on configuring and administering multiplexor services in *Messaging Server System Administrator's Guide*.

Note
To configure HTTP user mailboxes (for example, Messenger Express), see the topic on configuring and administering multiplexor services in *Messaging Server System Administrator's Guide*.

Chapter 4. Messaging Server Configuration

Oracle Communications Messaging Server Configuration

Starting with Messaging Server 7 Update 4, reference information about the `configutil` parameters is automatically generated and provided on the MsgServerDocWiki.

Chapter 5. Messaging Server Supported Standards

Supported Standards

See Supported Standards.

Chapter 6. MeterMaid Reference

MeterMaid Reference

This document contains MeterMaid reference information and contains the following sections:

- `configutil` Parameters
- Table Types
- `check_metermaid.so` Reference

`configutil` Parameters

For MeterMaid-specific configutil parameters, you can search for parameters containing `metermaid` in the configutil Parameters table.

Table Types

There are two possible table types allowed by the `metermaid.table.*.type` parameters:

- `greylisting` Tables
- `simple` Tables
- `throttle` Tables

`greylisting` Tables

New in Messaging Server 7 Update 4, greylisting tables may be used to provide an anti-spam/anti-virus technique. For more information about setting up these tables, see the topic on implementing greylisting by using MeterMaid in *Messaging Server System Administrator's Guide*.

`simple` Tables

Simple tables are new in Messaging Server 7 Update 2. A simple table may be used to store arbitrary data referenced by a key. The key data type is defined by `metermaid.table.tablename.data_type`, and the value data type is defined by `metermaid.table.tablename.value_type`. Some operations are only available to those simple tables where the value data type is `integer`.

`throttle` Tables

Throttle tables are used to specify a particular "hit count" `quota` over `quota_time` seconds to limit connections, transactions, or certain other components of incoming connections. MeterMaid automatically maintains the count over time, decrementing it back down after `quota_time` has passed.

`check_metermaid.so` Reference

The `check_metermaid.so` shared library is traditionally used to throttle incoming connections in a mapping table such as `PORT_ACCESS` or `MAIL_ACCESS`. New in Messaging Server 7 Update 2 is a set of new routines to use data stored in `simple` tables (please refer to the `metermaid.table.*.type` configutil parameter). These new routines now permit one to store, retrieve, and test arbitrary data stored in MeterMaid's ephemeral data store. Messaging Server 7 Update 4 adds the `greylisting` routine that works with `greylisting` tables.

The following table shows the routines available in `check_metermaid.so`, which of the two table types, simple and/or throttle, are supported for those routines, and a brief description of each one. Below that, detailed information about each routine is provided.

Note
If any error should occur during processing such as a failure to communicate with MeterMaid, or if invalid parameters are provided to these routines, the routines will simply return FALSE and the shared library callout will fail.

Routine	Description
`adjust`	Adds to or subtracts from an integer value in a table
`adjust_and_test`	Performs an adjust and then returns the result of a test operation
`fetch`	Returns a value from the table
`greylisting`	Returns TRUE if we are temporarily rejecting this transaction
`remove`	Removes an entry from a table
`store`	Stores a value into the table
`test`	Tests an integer value with a simple comparison, returning TRUE or FALSE
`throttle`	Increments a "hit count" and returns TRUE if `quota_count` is exceeded

The following sections provide a description of each routine, a table showing the parameters used by the routine, the value returned, if any, to the calling environment, and sample usage.

- `adjust` Routine
- `adjust_and_test` Routine
- `fetch` Routine
- `greylisting` Routine
- `remove` Routine
- `store` Routine
- `test` Routine
- `throttle` Routine

`adjust` Routine

Description

The `adjust` routine allows you to make a numeric adjustment to an integer value in a `simple` table.

Parameter	Description
table	Table in which *key* is found
key	Key corresponding to the value being adjusted
adjustment	Positive or negative value to be added to the value

Return Value
Returns TRUE with the new value for *key* after the adjustment has been made as the resultant string.

`adjust` works by taking the current integer value for *key* in the table called *table* and adding *adjustment* to it, then storing the resulting value back. *adjustment* can be negative, thus reducing the value which can be negative.

If *key* doesn't exist in *table*, it will be presumed to have an initial value of 0 and *key* will be stored into *table* with a new value of *adjustment*.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	
`simple`	`integer`	X
`simple`	`string`	
`throttle`	-	

Example

The following example shows that the current value for `fred@siroe.com` will be increased by 35 after the `adjust` is completed. If `fred@siroe.com` did not exist in the `scores` table, it would be stored with a value of 35.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,adjust,scores,fred@siroe.com,
```

`adjust_and_test` Routine

Description

The `adjust_and_test` routine allows you to make a numeric adjustment to an `integer` value in a `simple` table, and then test that value against a provided comparator.

Parameter	Description
table	Table in which *key* is found
key	Key corresponding to the value being adjusted
adjustment	Positive or negative value to be added to the value
comparator	Comparison symbol(s) '<', '>', and/or '=', followed by a numeric value

Return Value
Returns TRUE if the comparison is true, and FALSE otherwise; no resultant string is returned.

`adjust_and_test` first takes the current integer value for *key* in the table called *table* and adding *adjustment* to it, storing the resulting value back. The routine then compares the resulting value against the *comparator*, returning the result.

If *key* doesn't exist in *table*, it will be presumed to have an initial value of 0 and *key* will be stored into *table* with a new value of *adjustment*, and the comparison will be made against *adjustment*.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	
`simple`	`integer`	X
`simple`	`string`	
`throttle`	-	

Example

The following example shows that the current value for `fred@siroe.com` will be decreased by 2 after the `adjust` is completed. If `fred@siroe.com` did not exist in the `scores` table, it would be stored with a value of -2. Then this new value is checked to see if it is greater than or equal to 20, returning TRUE if it is.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,adjust_and_test,scores,fred@s
```

`fetch` Routine

Description

The `fetch` routine retrieves a value from a `simple` table.

Parameter	Description
table	Table in which *key* is found
key	Key corresponding to the value being returned

Return Value
Returns TRUE if *key* exists and returns its value as the resultant string, otherwise returns FALSE.

`fetch` retrieves the value associated with *key* in *table* and returns it as the resultant string. If *key* does not exist in *table*, then FALSE is returned and no resultant string is available.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	X
`simple`	`integer`	X
`simple`	`string`	X
`throttle`	-	

Example

The following example retrieves the current score for `fred@siroe.com` which can then use that information for subsequent processing, such as in a mapping table.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,fetch,scores,fred@siroe.com]
```

`greylisting` Routine

The `greylisting` routine is used to validate entries in the table based on time and resubmission attempts. This is used as part of a greylisting setup. For more information, see the topic on implementing greylisting by using MeterMaid in *Messaging Server System Administrator's Guide*.

Parameter	Description
table	Table in which *key* is checked/stored
key	Key corresponding to the value being tested for greylisting

Return Value	
Returns TRUE if this probe should cause a temporary rejection for the submission attempt, otherwise returns FALSE to permit the attempt.	`greylisting` first probes *table* for *key* to see whether this entry has been previously permitted. If it is found to be allowed, FALSE is returned to permit the submission. Then it checks to see whether *key* exists and is in its resubmission period (specified by the `resubmit_time` table option). If so, *key* is marked as valid and is permitted (returning FALSE). If *key* exists, but is in the `block_time` period, the submission is refused as validation occurs after `block_time` has passed, and `greylisting` returns TRUE to return a temporary rejection for the attempt. Lastly, if *key* does not exist, it is stored into *table* and `greylisting` returns TRUE to return a temporary rejection for this new attempt.

Example

This example shows a greylisting probe for mail from barney@example.com going to local user fred@siroe.com. If this returns TRUE, then the mapping code should return a temporary rejection so that the message submission should be reattempted later.

```
$[IMTA_LIB:check_metermaid.so,greylisting,greylist_table,192.168.10.34|barney@e
```

`remove` Routine

Description

The `remove` routine removes an entry from table.

Parameter	Description
table	Table in which *key* is found
key	Key corresponding to the value being removed

Return Value
Returns TRUE if *key* was removed from *table*, otherwise returns FALSE.

When an entry in a table is no longer needed, it may be removed using `remove`. Subsequent attempts to access *key* will result in its value not being found.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	X
`simple`	`integer`	X
`simple`	`string`	X
`throttle`	-	X

Example

The following example can be used when the record for `fred@siroe.com` is no longer needed.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,remove,scores,fred@siroe.com]
```

`store` Routine

Description

The `store` routine is used to store a new value into the table.

Parameter	Description
table	Table into which the new value is to be stored
key	Key corresponding to the value
value	New value

Return Value
Returns TRUE if the new value was stored, FALSE otherwise. Returns no resultant string.

`store` is similar to `adjust` in that it can be used to put data into a table. Unlike `adjust`, however, any previous value that may exist is overwritten by `store`. Also, in addition to integer data, strings may also stored into those tables that permit it. For a `greylisting` table, *value* is ignored and instead the new *key* is stored into *table* as a valid entry for subsequent queries.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	X
`simple`	`integer`	X
`simple`	`string`	X
`throttle`	-	

Example

The following example sets an initial value into a table that can be used by subsequent `fetch` operations.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,store,loginhosts,barney@siroe
```

`test` Routine

Description

The `test` routine allows you to compare an integer value in a `simple` table against a supplied comparator.

Parameter	Description
table	Table in which *key* is found
key	Key corresponding to the value being adjusted
comparator	Comparison symbol(s) '<', '>', and/or '=', followed by a numeric value

Return Value
Returns TRUE if the comparison is true, and FALSE otherwise; no resultant string is returned.

`test` takes the current integer value for *key* in the table called *table* and compares the resulting value against the *comparator*, returning the result. If *key* does not exist in *table*, then 0 is used as the value to be compared.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	X
`simple`	`integer`	X
`simple`	`string`	
`throttle`	-	X

Example

The following example tests the number of login attempts made to see whether it exceeds a defined threshold.

```
$[/opt/sun/comms/messaging/lib/check_metermaid.so,test,logins,wilma@siroe.com,>
```

`throttle` Routine

Description

The `throttle` routine is used to count incoming connections or transactions over a period of time enforcing a quota limit.

Parameter	Description
table	Table in which is holding the items being counted
key	Key corresponding to the particular "hit count" to be incremented

Return Value
Returns TRUE if `quota` has been exceeded during the past `quota_time` seconds, otherwise returns FALSE.

For more detailed information on setting up `throttle` tables with configuration examples, refer to Limit Excessive IP Address Connections Using Metermaid - Example.

Supported Table and Value Types

Table	Value	Supported?
`greylisting`	-	
`simple`	`integer`	
`simple`	`string`	
`throttle`	-	X

Chapter 7. SMS Channel Option File

SMS Channel Option File

The Messaging Server SMS (Short Message Service) channel is a one-way email to SMS gateway. Mail can be sent to an SMS gateway, but handling of SMS notifications (that is, replies and delivery receipts) and origination of email from SMS users (mobile to email) is presently not supported. The channel converts enqueued email messages to SMS messages. This conversion process includes handling of multipart MIME messages as well as character set translation issues.

The generated SMS messages are submitted to a Short Message Service Centre (SMSC) using the Short Message Peer to Peer (SMPP) protocol. Specifically, SMPP V3.4 is used over a TCP/IP connection to the SMSC's SMPP server. Operating in this capacity, the channel functions as a External Short Message Entity (ESME).

For more information about the SMS channel, see *Messaging Server System Administrator's Guide*.

An option file may be used to control various characteristics of the SMS channel. The channel options are stored in a text file in the *msg-svr-base*`/config/` directory. The name of the file takes the form:

channel-name`_option`

For instance, if the channel is named `sms_mway`, then the channel option file is:

msg-svr-base`/config/sms_mway_option`

Format of the File

Each option is placed on a single line in the file using the format:

option-name=option-value

For example:

```
PROFILE=GSM
SMSC_DEFAULT_CHARSET=iso-8859-1
USE_UCS2=1
```

A sample option file named `sms_option.sample` is distributed with Messaging Server. Copy this option file and use it as a starting point.

Available Options

The SMS channel contains a number of options which divide into four broad categories: email to SMS conversion, SMS fields, SMPP protocol, and localization. These categories and their corresponding options are detailed in the following sections.

Email to SMS Conversion

The email to SMS conversion options control the email to SMS conversion process. In general, a given email message may be converted into one or more SMS messages. These options are described in Table 4-29.

Table 4-29 SMS Channel Options: Email to SMS Conversion

Option	Description
`GATEWAY_NOTIFICATIONS`	Specify whether or not to convert email notification messages to SMS messages. Default: `0`
`LOG_PAGE_COUNT (0,1,2)`	The `LOG_PAGE_COUNT` SMS channel option only takes effect when logging is enabled for the channel with the `logging` channel keyword. When logging is enabled, this option controls the value recorded in the `mail.log` file's message size field. Normally that field gives the block size of the underlying message file. When `LOG_PAGE_COUNT` has a non-zero value, the number of transmitted pages will instead be recorded in that field of the log file. 0 - Log the block size of the underlying message file. This is the default behavior when `LOG_PAGE_COUNT` is not specified. 1 - Log the count of pages sent when the entire message is successfully transmitted to the recipient. Otherwise, log a page count of zero even if some pages were sent to the recipient. 2 - Log the count of pages sent to the recipient regardless of whether or not the entire message was sent. The distinction between `LOG_PAGE_COUNT=1` and `LOG_PAGE_COUNT=2` is only relevant when a message is sufficiently large that it will be transmitted as several pages. In that case, it is possible that before transmitting all the pages an error might occur. For example, the network between the MTA and the remote SMPP server goes down. In that case, a later retransmission attempt will be made for the message. For each attempt, the previously sent pages are sent again along with the pages which were not sent. Sites may choose whether or not they want to record the count of pages successfully sent during these failed delivery attempts.
`MAX_MESSAGE_PARTS` (Integer)	Maximum number of message parts to extract from an email message. When converting a multi-part email message to an SMS message, only the first `MAX_MESSAGE_PARTS` text parts will be converted. The remaining parts are discarded. By default, `MAX_MESSAGE_PARTS` is 2. To allow an unlimited number of message parts, specify a value of -1. When a value of 0 is specified, then no message content will be placed into the SMS message. This has the effect of using only header lines from the email message (for example, `Subject:`) to generated the SMS message. Note that an email message containing both text and an attachment will typically consist of two parts. Note further that only message parts of type text are converted. All other MIME content types are discarded.

`MAX_MESSAGE_SIZE` (Integer, >=10)	Maximum number of bytes to extract from an email message. With this option, an upper limit may be placed on the total number of bytes placed into the SMS messages generated from an email message. Specifically, a maximum of `MAX_MESSAGE_SIZE` bytes will be used for the one or more generated SMS messages. Any additional bytes are discarded. By default, an upper limit of 960 bytes is imposed. This corresponds to `MAX_MESSAGE_SIZE`=960. To allow any number of bytes, specify a value of zero. The count of bytes used is made after converting the email message from Unicode to either the SMSC's default character set or UCS2. This means, in the case of UCS2, that a `MAX_MESSAGE_SIZE` of 960 bytes will yield, at most, 480 characters since each UCS2 character is at least two bytes long. Note that the `MAX_MESSAGE_SIZE` and `MAX_PAGES_PER_MESSAGE` options both serve the same purpose: to limit the overall size of the resulting SMS messages. For example, `MAX_MESSAGE_SIZE=960` and `MAX_PAGE_SIZE=160` implies `MAX_PAGES_PER_MESSAGE=6`. The two different options exist to allow control of the overall size or number of pages without having to consider the maximal size of a single SMS message, `MAX_PAGE_SIZE`. While this may not be important in the channel option file, it is important when using the `MAXPAGES` or `MAXLEN` addressing attributes described in the http://docs.sun.com/doc/819-4428. Finally, note that the smaller of the two limits of `MAX_MESSAGE_SIZE` and `MAX_PAGE_SIZE * MAX_PAGES_PER_MESSAGE` is used.
`MAX_PAGE_SIZE` (Integer, >=10)	Maximum number of bytes to allow into a single SMS message. By default, a value of 160 bytes is used.
`MAX_PAGES_PER_MESSAGE` (Integer, 1-255)	Maximum number of SMS messages to generate for a given email message. This option truncates the email message, only converting to SMS messages that part of the email message which fits into `MAX_PAGES_PER_MESSAGE` SMS messages. By default, `MAX_PAGES_PER_MESSAGE` is set to the larger of 1 or `MAX_MESSAGE_SIZE` divided by `MAX_PAGE_SIZE`.
`SMSC_DEFAULT_CHARSET` (string)	Default character set used by the SMSC. The character set names in the *msg-svr-base*`/imta/confic/charsets.txt` file are used. US-ASCII is the default. When processing an email message, the header lines and text message parts are first decoded and then converted to Unicode. Next, the data is converted to either the SMCS's default character set or USC2, as follows: • `1` --The SMSC default character set is used whenever possible. When the originating email message contains glyphs not in the SMSC default character set, then the UCS2 character set is used.* `0` --The SMSC default character set is always used. Glyphs nor available in that character set are represented by mnemonics (for example, "AE" for AE-ligature).

USE_HEADER_FROM	Set this option to allow the From: address to be passed to the SMS channel. The value indicates where the From: address is taken from and what format it will have. 0 - SMS source address never set from the From: address. Use attribute-value pair found 1 - SMS source address set to from-local@from-domain, where the From: address is: @from-route:from-local@from-domain 2 - SMS source address set to from-local, where the From: address is: @from-route:from-local@from-domain Default: 0{}
USE_HEADER_PRIORITY (0 or 1)	Controls the use of priority information from the email message's header (RFC822 Priority: header lines). By default, information from the Priority: header line is used to set the resulting SMS message's priority flag, overriding the default SMS priority specified with the DEFAULT_PRIORITY option. This case corresponds to USE_HEADER_PRIORITY=1. To disable use of the RFC822 Priority: header line, specify USE_HEADER_PRIORITY=0. The default is USE_HEADER_PRIORITY =1. See the description of the DEFAULT_PRIORITY option for further information on the handling the SMS priority flag.
USE_HEADER_REPLY_TO (0 or 1)	Controls the use of Reply-to: header lines when generating SMS source addresses. When SET_SMS_SOURCE_ADDRESS=1, this option controls whether or not a Reply-to: or Resent-reply-to: header line is considered for use as the SMS source address. By default, Reply-to: and Resent-reply-to: header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1. Note that RFC 2822 has deprecated the use of Reply-to: and Resent-reply-to: header lines. This is the reason why, by default, USE_HEADER_REPLY_TO=0.
USE_HEADER_RESENT (0 or 1)	Controls the use of Resent-*: header lines when generating originator information. When SET_SMS_SOURCE_ADDRESS=1, this option controls whether or not Resent- header lines are considered for use as the SMS source address. By default, Resent- header lines are ignored. This corresponds to an option value of 0. To enable consideration of these header lines, use an option value of 1. Note that RFC 2822 has deprecated the use of Resent- header lines; hence, why this option has the default value of 0.
USE_HEADER_SENSITIVITY (0 or 1)	Controls the use of privacy information from the email message's header (RFC822 Sensitivity: header lines).By default, information from the Sensitivity: header line is used to set the resulting SMS message's privacy flag, overriding the default SMS privacy specified with the DEFAULT_PRIVACY option. This case, which is the default, corresponds to USE_HEADER_SENSITIVITY=1. To disable use of RFC822 Sensitivity: header lines, specify USE_HEADER_SENSITIVITY=0. See the description of the DEFAULT_PRIVACY option for further information on the handling the SMS privacy flag.

USE_UCS2 (0 or 1)	Specifies that the UCS2 character set is to be used in SMS messages when applicable. The default behavior is to use the UCS2 character set, and corresponds to USE_UCS2=1. To disable the use of the UCS2 character set, specify USE_UCS2=0. See the description of the SMSC_DEFAULT_CHARSET option for further information on character set issues.

SMS Gateway Server Option

The SMS Gateway Server Option specifies the Gateway profile. shows. This option is described in Table 4-30.

Table 4-30 SMS Channel Options: SMS Gateway Server Option

Option	Description
GATEWAY_PROFILE	Match the gateway profile name configured in the SMS Gateway Server's configuration file, sms_gateway.cnf.
ROUTE_TO (string, IP hostname, 1-64 bytes)	All SMS messages targeted to the profile will be rerouted to the specified IP host name using an email address of the form: SMS-destination-address@route-to where SMS-destination-address is the SMS message's destination address and the route-to is the IP host name specified with this option. The entire content of the SMS message is sent as the content of the resulting email message. The PARSE_RE_* options are ignored. ----**Note --** Use of PARSE_RE_* and ROUTE_TO options are mutually exclusive. Use of both in the same gateway profile is a configuration error.

SMS Fields

The SMS fields options control SMS-specific fields in generated SMS messages. These options are described in Table 4-31.

Table 4-31 SMS Channel Options: SMS Fields

Option	Description

`DEFAULT_DESTINATION_NPI` (Integer, 0-255)	Default NPI for SMS destination addresses. By default, destination addresses are assigned an NPI (Numeric Plan Indicator) value of zero. With this option, an integer value in the range `0` to `255` may be assigned. Typical NPI values include: `0` - Unknown `1` - ISDN (E.163, E.164) `3` - Data (X.121) `4` - Telex (F.69) `6` - Land Mobile (E.212) `8` - National `9` - Private `10` - ERMES `14` - IP address (Internet) `18` - WAP client ID `>=19` - Undefined Values for this option may be specified in one of three ways: • A decimal value (for example, `10`). • A hexadecimal value prefixed by `0x` (for example, `0x0a`). • One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): data (`3`), default (`0`), e.163 (`1`), e.212 (`6`), ermes (`10`), f.69 (`4`), internet (`14`), ip (`14`), isdn (`1`), land-mobile (`6`), national (`8`), private (`9`), telex (`4`), unknown (`0`), wap (`18`), x.121 (`3`).
`DEFAULT_DESTINATION_TON` (Integer, 0-255)	Default TON for SMS destination addresses. By default, destination addresses are assigned a TON (Type of Number) designator value of zero. With this option, an alternate integer value in the range of `0` to `255` may be assigned. Typical TON values include: `0` - Unknown `1` - International `2` - National `3` - Network- specific `4` - Subscriber number `5` - Alphanumeric `6` - Abbreviated `>=7` - Undefined Values for this option may be specified in one of three ways: • A decimal value (for example, `10`). • A hexadecimal value prefixed by `0x` (for example, `0x0a`). • One of the following case-insensitive text strings (the associated decimal value is shown in parentheses): abbreviated (`6`), alphanumeric (`5`), default (`0`), international (`1`), national (`2`), network-specific (`3`), subscriber (`4`), unknown (`0`).

`DEFAULT_PRIORITY` (Integer, 0-255)	Default priority setting for SMS messages. All SMS messages have a mandatory priority field. The interpretation of SMS priority values is described in Table 4-32. With this option, the default priority to assign to SMS messages may be specified. When not specified, a default priority of `0` is used for `PROFILE=GSM` and CDMA, and a priority of 1 for `PROFILE=TDMA`. Note that if `USE_HEADER_PRIORITY=1` and an email message has an RFC822 `Priority:` header line, then the priority specified in that header line is instead used to set the priority of the resulting SMS message. Specifically, the results are as follows: `0` --The SMS priority flag is always set in accord with the `DEFAULT_PRIORITY` option. The RFC822 Priority: header line is always ignored. `1` (default) --The originating email message's RFC822 `Priority:` header line is used to set the SMS message's priority flag. If that header line is not present, then the SMS priority flag is set using the `DEFAULT_PRIORITY` option. In translating RFC822 `Priority:` header line values to SMS priority flags, the mappings used are described in Table 4-33.
`DEFAULT_PRIVACY` (Integer, -1, 0-255)	Default privacy value flag for SMS messages. Whether or not to set the privacy flag in an SMS message, and what value to use is controlled by the `DEFAULT_PRIVACY` and `USE_HEADER_SENSITIVITY` options. By default, a value of -1 is used for `DEFAULT_PRIVACY`. The results from the combination of `DEFAULT_PRIVACY` and `USE_HEADER_SENSITIVITY` values are described in Table 4-34. The SMS interpretation of privacy values is as follows: • 0 - Unrestricted • 1 - Restricted • 2 - Confidential • 3 - Secret • >=4 - Undefined To translate Sensitivity: header line values to SMS privacy values, the following mapping is used: • Personal - 1 (Restricted • Private - 2 (Confidential)
`DEFAULT_SERVICE_TYPE` (String, 055 bytes)	SMS application service associated with submitted SMS messages. By default, no service type is specified (that is, a zero length string). Some common service types are: CMT (cellular messaging), SPT (cellular paging), VMN (voice mail notification), VMA (voice mail alerting), WAP (wireless application protocol), and USSD (unstructured supplementary data services).

DEFAULT_SOURCE_ADDRESS (String, 0-20 bytes)	Default SMS source address to use for SMS messages generated from email messages. Note that the value specified with this option is typically overridden by the email message's originator address when SET_SMS_SOURCE_ADDRESS=1. The default is no source address is specified (0 length string).
DEFAULT_SOURCE_NPI (Integer, 0-255)	Default NPI for SMS source addresses. By default, source addresses are assigned an NPI value of zero. With this option, an alternate integer value in the range of 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_NPI option for a list of typical NPI values.
DEFAULT_SOURCE_TON (Integer, 0-255)	Default TON for SMS source addresses. By default, source addresses are assigned a TON designator value of zero. With this option, an alternate integer value in the range of 0 to 255 may be assigned. See the description of the DEFAULT_DESTINATION_TON option for a list of typical TON values.
DEFAULT_VALIDITY_PERIOD (String, 0-252 bytes)	Default validity period for SMS messages. This option specifies a different relative validity period. By default, SMS messages are given no relative validity period, using the SMSC's default value. Values may be specified in units of seconds, minutes, hours, or days: nnn - Implicit units of seconds; for example, 604800nnns - Units of seconds; for example, 604800snnnm - Units of minutes; for example, 10080mnnnh - Units of hours; for example, 168hnnnd - Units of days, for example, 7d A specification of 0, 0s, 0m, 0h, or 0d may be used to select the SMSC's default validity period. That is, when a specification of 0, 0s, 0m, 0h, or 0d is used, an empty string is specified for the validity period in generated SMS messages. Note that this option does not accept values in UTC format.
DEFAULT_ADDRESS_NUMERIC (0 or 1)	Reduce the destination SMS address to only the characters 0-9. This option strips all non-numeric characters from the SMS destination address extracted from the email envelope To: address. For instance, if the envelope To: address is: "(800) 555-1212"@sms.siroe.com then it will be reduced to:8005551212@sms.siroe.com To enable this stripping, specify a value of 1 for this option. By default, this stripping is disabled which corresponds to an option value of 0. Note that when enabled, the stripping is performed before any destination address prefix is added via the DESTINATION_ADDRESS_PREFIX option.
DESTINATION_ADDRESS_PREFIX (String)	Text string with which to prefix destinationSMS addresses. In some instances, it may be necessary to ensure that all SMS destination addresses are prefixed with a fixed text string; for example, "+". This option may be used to specify just such a prefix. The prefix will then be added to any SMS destination address which lacks the specified prefix. To prevent being stripped by the DESTINATION_ADDRESS_NUMERIC option, this option is applied after the DESTINATION_ADDRESS_NUMERIC option.

PROFILE (String)	Specifies the SMS profile to use with the SMSC. Possible values are GSM, TDMA, and CDMA. When not specified, GSM is assumed. This option is only used to select defaults for other channel options such as DEFAULT_PRIORITY and DEFAULT_PRIVACY.
SET_SMS_SOURCE_ADDRESS (0 or 1)	Set the SMS source address to the originator address of the email message. Use of this option forces the SMS source address TON to be set to alphanumeric (0x05), and the SMS source address to be an originator address extracted from the email message. As email messages may have a number of originator addresses, the particular address chosen is the one most likely to be the address to which any replies should be directed. Consequently, the choice is made from one of the seven header lines described in Table 4-35, listed in order of decreasing preference: The selected address is reduced to just its local and domain parts; that is, any source route, phrase, or comments are stripped from the address. Furthermore, if the length of the reduced address exceeds 20 bytes, it will be truncated to 20 bytes. When none of the seven listed header lines are suitable, the default source SMS address is instead used as specified with the DEFAULT_SOURCE_ADDRESS option. In that case, the TON is set as per the DEFAULT_SOURCE_TON. To enable this option, specify SET_SMS_SOURCE_ADDRESS=1. By default, this option is enabled.
USE_SAR (0 or 1)	Sequence multiple SMS messages using the SMS sar_fields. Sufficiently large email messages may need to be broken into multiple SMS messages. When this occurs, the individual SMS messages can optionally have sequencing information added using the SMS sar_ fields. This produces a "segmented" SMS message which can be re-assembled into a single SMS message by the receiving terminal. Specify USE_SAR=1 to indicate that this sequencing information is to be added when applicable. The default is to not add sequencing information and corresponds to USE_SAR=0. When USE_SAR=1 is specified, the REVERSE_ORDER option is ignored.

Table 4-32 describes the interpretation of the priority field for the DEFAULT_PRIORITY option.

Table 4-32 Priority Fields for DEFAULT_PRIORITY

Value	GSM	TDMA	CDMA
0	Non-priority	Bulk	Normal
1	Priority	Normal	Interactive
2	Priority	Urgent	Urgent
3	Priority	Urgent	Emergency

Table 4-33 describes the mappings used in translating Priority: header line values to SMS priority flags for the DEFAULT_PRIORITY option.

Table 4-33 Mappings for Priority Flags

RFC822	SMS Priority Flag		
Priority: value	**GSM**	**TDMA**	**CDMA**
Third	Non-priority (0)	Bulk (0)	Normal (0)
Second	Non-priority (0)	Bulk (0)	Normal (0)
Non-urgent	Non-priority (0)	Bulk (0)	Normal (0)
Normal	Non-priority (0)	Normal (1)	Normal (0)
Urgent	Priority (1)	Urgent (2)	Urgent (2)

The results from the combination of `DEFAULT_PRIVACY` and `USE_HEADER_SENSITIVITY` values are described in Table 4-34.

Table 4-34 Results from DEFAULT_PRIVACY and USE_HEADER_SENSITIVITY Values

DEFAULT_PRIVACY	USE_HEADER_SENSITIVITY	Result
1	0	The SMS privacy flag is never set in SMS messages.
n >=0	0	The SMS privacy flag is always set to the value n. RFC822 Sensitivity: header lines are always ignored.
-1 (default	1 (default)	The SMS message's privacy flag is only set when the originating email message has an RFC822 `Sensitivity:` header line. In that case, the SMS privacy flag is set to correspond to the `Sensitivity:` header line's value. This is the default.
n >= 0	1	The SMS message's privacy flag is set to correspond to the originating email message's RFC822 `Sensitivity:` header line. If the email message does not have a Sensitivity: header line, then the value of the SMS privacy flag is set to n.

Table 4-35 describes the seven header lines used with the `SET_SMS_SOURCE_ADDRESS` option, their restrictions and SMS source address TON (if applicable) in decreasing preference.

Table 4-35 SET_SMS_SOURCE_ADDRESS Header Restrictions

Email message field	Restrictions	TON
1. Resent-reply-to:	Requires `USE_HEADER_RESENT=1` and `USE_HEADER_REPLY_TO=1`	
2. Resent-from:	Requires `USE_HEADER_RESENT=1`	
3. Reply-to:	Requires `USE_HEADER_REPLY_TO=1`	0x05
4. From:		
5. Resent-sender:	Requires `USE_HEADER_RESENT=1`	
6. Sender:		
7.Envelope From:		
8. `DEFAULT_SOURCE_ADDRESS`	Used as a last resort (that is, when the envelope From: address is empty)	As per `DEFAULT_SOURCE_TON`

SMPP Protocol

The SMPP protocol options are associated with the use of the SMPP protocol over TCP/IP. The options with names beginning with the string "`ESME_`" serve to identify the MTA when it acts as an External Short Message Entity (ESME); that is, when the MTA binds to an SMPP server in order to submit SMS messages to the server's associated SMSC. These options are described in Table 4-36.

Table 4-36 SMS Channel Options: SMPP Protocol

Option	Description
`ESME_ADDRESS_NPI` (Integer, 0-255)	ESME NPI to specify when binding to the SMPP server. By default, bind operations specify an ESME NPI value of zero indicating an unknown NPI. With this option, an alternate integer value in the range `0` to `255` may be assigned. See the description of the `DEFAULT_DESTINATION_NPI` option for a table of typical NPI values.
`ESME_ADDRESS_TON` (Integer, 0-255)	ESME TON to specify when binding to the SMPP server. By default, bind operations specify an ESME TON value of `0`. With this option, an alternate integer value in the range `0` to `255` may be assigned. See the description of the `DEFAULT_DESTINATION_TON` option for a table of typical TON values.
`ESME_IP_ADDRESS` (String, 0-15 bytes)	IP address of the host running Messaging Server. When binding to the SMPP server, the bind PDU indicates that the client's (that is, ESME's) address range is an IP address. This is done by specifying a TON of `0x00` and an NPI of `0x0d`. The value of the address range field is then set to be the IP address of the host running the SMS channel. Specify the IP address in dotted decimal format; for example, `127.0.0.1`.
`ESME_PASSWORD` (String, 0-9 bytes)	Password to present when binding to the SMPP server. If a password is required, then specify it with this option. By default, a zero-length password string is presented.
`ESME_SYSTEM_ID` (String, 0-15 bytes)	System identification to present to the SMSC when binding. If a password is required, then specify it with this option. By default, a zero-length password string is presented.
`ESME_SYSTEM_TYPE` (String, 0-12 bytes)	System type for the MTA to present to the SMSC when binding. By default, no system type is specified (that is, a zero-length string is used).

MAX_PAGES_PER_BIND (Integer, >=0)	Maximum number of SMS messages to submit during a single session with an SMPP server. Some SMPP servers may limit the maximum number of SMS messages submitted during a single, bound session. In recognition of this, this option allows specification of the maximum number of SMS messages to submit during a single session. Once that limit is reached, the channel unbinds, closes the TCP/IP connection, re-connects, and then rebinds. By default, a value of 1024 is used for MAX_PAGES_PER_BIND. Note that the channel also detects ESME_RTHROTTLED errors and adjusts MAX_PAGES_PER_BIND during a single run of the channel accordingly.
REVERSE_ORDER (0 or 1)	Transmission sequence of multi-part SMS messages. When an email message generates more than one SMS message, those SMS messages can be submitted to the SMSC in sequential order (REVERSE_ORDER=0), or reverse sequential order (REVERSE_ORDER=1). Reverse sequential order is useful for situations where the receiving terminal displays the last received message first. In such a case, the last received message will be the first part of the email message rather than the last. By default, REVERSE_ORDER=1 is used. Note that this option is ignored when USE_SAR=1 is specified.
SMPP_MAX_CONNECTIONS (Integer, 1-50)	Maximum number of simultaneous SMPP server connections per process. As each connection has an associated thread, this option also places a limit on the maximum number of "worker" threads per process. By default, SMPP_MAX_CONNECTIONS=20.
SMPP_PORT (Integer, 1-65535)	TCP port on which the SMPP server listens. The TCP port may be specified with either this option or the port channel keyword. This port number must be specified through either of these two mechanisms. If it is specified with both mechanisms, then the setting made with the SMPP_PORT option takes precedence. Note that there is no default value for this option.
SMPP_SERVER (String, 1-252 bytes)	Host name of the SMPP server to which to connect. By default, the IP host name of the SMPP server to which to connect is the official host name associated with the channel; that is, the host name shown on the second line of the channel's definition in MTA's configuration. This option may be used to specify a different host name or IP address which overrides that specified in the channel definition. When specifying an IP address, use dotted decimal notation; for example, 127.0.0.1
TIMEOUT (Integer, >=2)	Timeout for completion of read and write actions with the SMPP server. By default, a timeout of 30 seconds is used when waiting for data "writes" to the SMPP server to complete or for data to be received from the SMPP server. Use the TIMEOUT option to specify, in units of seconds, a different timeout value. The specified value should be at least 2 seconds.

`[VENDOR_CODES=PERM]`	By default, the SMS channel treats any unrecognized SMPP response code as a temporary failure. This includes response codes in the range 0x400 through 0x4FF. However, the SMPP specification has reserved that range for private use by SMSCs. It is, therefore, possible for an SMSC to have private response codes in that range which should be treated as permanent errors rather than as temporary errors. In that case, the SMS channel needs to be configured so as to recognize these private, permanent error codes. This is done in the option file using this `[VENDOR_CODES=PERM]` section. Each entry in this section takes one of three forms:
	
	`code=description`
	`code1,code2,code3,..=description`
	`code1-code2=description`
	
	The code is the numeric value of the error code and may be a decimal, octal, or hexadecimal number. Octal numbers start with a zero and then contain only octal digits, 0-7; whereas, hexadecimal numbers start with "0x" or "0X" and contain hexadecimal digits, 0-9 and a-f or A-F. For example:
	
	`[VENDOR_CODES=PERM]`
	`0x400,0x408=Mailbox Moved`
	`0x401=Recipient not allowed`
	`0x4a0-0x4af,0x4fe=Unknown permanent error`
	
	Note that it is possible to specify codes outside the range 0x400 - 0x4FF. This is permitted and allows sites to treat some temporary errors as permanent errors.

Localization

The Localization options allow for localization of text fields inserted into SMS messages. These options are described in Table 4-37. In constructing SMS messages, the SMS channel has a number of fixed text strings it places into those messages. These strings, for example, introduce the email's From: address and Subject: header line. With the channel options described below, versions of these strings may be specified for different languages and a default language for the channel then specified. This section of the option file appears as follows:

```
LANGUAGE=default-language
```

```
[language=i-default]FROM_PREFIX=From:SUBJECT_PREFIX=Subj:
CONTENT_PREFIX=Msg:LINE_STOP= NO_MESSAGE=[no message]REPLY_PREFIX=Re:
```

```
[language=en]FROM_PREFIX=From:SUBJECT_PREFIX=Subj:
CONTENT_PREFIX=Msg:LINE_STOP= NO_MESSAGE=[no message]REPLY_PREFIX=Re: ...
```

Within each `[language=x]` block, the localization options relevant to that language may be specified. If a particular option is not specified within the block, then the global value for that option is used. A localization option specified outside of a `[language=x]` block sets the global value for that option.

For the options listed below, the string values must be specified using either the US-ASCII or UTF-8 character sets. Note that the US-ASCII character set is a special case of the UTF-8 character set.

Table 4-37 SMS Channel Options: Localization

Option	Description
`CONTENT_PREFIX` (String, 0-252 bytes)	Text string to place in the SMS message before the content of the email message itself. Default global value is the US-ASCII string "`Msg:`".
`DSN_DELAYED_FORMAT`	Formatting string for delivery delay notifications
`DSN_FAILED_FORMAT`	Formatting string for delivery failure notifications
`DSN_RELAYED_FORMAT`	Formatting string for relay notifications.
`DSN_SUCCESS_FORMAT`	Formatting string to successful delivery notifications.
`FROM_FORMAT` (String, 0-252 bytes)	Text to display indicating the originator of the email message. The default global value is the US-ASCII string "`$a`" which substitutes in the originator's email address.
`FROM_NONE` (String, 0-252 bytes)	Text to display when there is no originator address to display. The default global value is an empty string. Note that normally, this option is never used as sites typically reject email messages which lack any originator address.
`LANGUAGE` (String, 0-40 bytes)	Language group from which to select text fields. If not specified, the language is derived from the host's default locale specification. If the host's locale specification is not available or corresponds to "C" then i-default is used. (i-default corresponds to "English text intended for an international audience.")
`LINE_STOP` (String, 0-252 bytes)	Text to place at the end of each line extracted from the email message. The default global value is the US-ASCII space character.
`NO_MESSAGE` (String, 0-252 bytes)	Text to indicate that the message contains no content. The default global value is the US-ASCII string "`[no message]`".
`REPLY_PREFIX` (String, 0-252 bytes)	Reserved for future use. The default global value is the US-ASCII string "`Re:` ".
`SUBJECT_FORMAT` (String, 0-252 bytes)	Formatting template to format the content of the Subject: header line for display in the SMS message. The global default value for this option is the US-ASCII string "(`$s`)".See the `SUBJECT_NONE` option for a description of the handling when there is no Subject: header line or the content of that header line is an empty string.
`SUBJECT_NONE` (String, 0-252 bytes)	Text to display when no subject exists for the email message, or the Subject: header line's value is an empty string. The default global value for this option is the empty string.

Miscellaneous

`Debug`: Enable verbose debug output.

Chapter 8. Tailor File

Tailor File

The MTA tailor file (`imta_tailor`) is an option file in which the location of various MTA components are set. This file must always exist in the `<msg-svr-base>/config` directory for the MTA to function properly. The file may be edited to reflect the changes in a particular installation. Some options in the file should not be edited. The MTA should be restarted after making any changes to the file. It is preferable to make the changes while the MTA is down.

Note
As part of a future project to support an XML configuration, the MTA's tailor file is in the process of being deprecated. Beginning with Messaging Server 7, only the following eight options are honored. Any other option specified in the tailor file is ignored, even if it is set to a non-default value.

An option setting has the form:

```

option=value
```

The value can be either a string or an integer, depending on the option's requirements. If you make changes to values in order to specify directory paths, note that these values are prefixes, not paths. You must include a trailing slash. Comments are allowed. Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored. Options that are available and can be edited are shown in the following table:

Tailor File Options

Option	Desription
`IMTA_LANG`	Locale of the MTA's notary messages. By default it is `<msg-svr-base>/imta/locale/C/LC_MESSAGES`.
`IMTA_RETURN_PERIOD`	Controls the return of expired messages and the generation of warnings. The default value for this option is `1`. If this options is set to an integer value `N`, then the associated action is only performed every `N` times the return job runs. By default, the return job runs once every day.
`IMTA_RETURN_CLEANUP_PERIOD`	The periodic return job also includes two hooks for executing site-supplied procedures. A site-supplied procedure to run prior to the usual return job functions may be supplied as `$SERVERROOT/data/site-programs/bin/pre_return`. Sites may provide their own clean up procedure, to run after the usual return job functions, as a `$SERVERROOT/data/site-programs/bin/daily_cleanup` shell script. The periodic return job will automatically execute such a procedure, if it exists. By default, it executes it each time it runs. This frequency may be controlled via the `IMTA_RETURN_CLEANUP_PERIOD` tailor file option. For example, if you set `IMTA_RETURN_CLEANUP_PERIOD=3`, then the daily_cleanup script will be run only every third time the return job runs. The default value for this option is `1`.
`IMTA_RETURN_SPLIT_PERIOD`	Controls splitting of the `mail.log` file. The default value for this option is `1`. If this options is set to an integer value `N`, then the associated action is only performed every `N` times the return job runs. By default, the return job runs once every day.
`IMTA_TMP`	Temporary file directory; defaults to `/tmp`. For Linux systems, `/tmp` is a disk file system rather than a `tmpfs` (temporary file system), so you should use `/dev/shm`.
`IMTA_USER`	Name of the postmaster. The default is `inetmail`. If this is changed be sure to edit the `<msg-svr-base>/config/aliases` file to reflect the change to the postmaster address.
`IMTA_USER_USERNAME`	Specifies the userid of the subsidiary account the MTA uses for certain "non-privileged" operations---operations which it doesn't want to perform under the usual MTA account. The default is `nobody`.
`IMTA_WORLD_GROUP`	Can perform certain privileged operations as a member of this group. The default is `mail`.